Benutzerhandbuch - Deutsch

FUJITSU

openUTM V6.0

Anwendungen generieren

Ausgabe Mai 2010

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an manuals@ts.fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2000

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2000 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright und Handelsmarken

Copyright © Fujitsu Technology Solutions GmbH 2010.

Alle Rechte vorbehalten.
Liefermöglichkeiten und technische Änderungen vorbehalten.

Alle verwendeten Hard- und Softwarenamen sind Handelsnamen und/oder Warenzeichen der jeweiligen Hersteller.

Inhalt

1 Einleitung 11

1.1 Zielgruppe und Konzept des Handbuchs 13

1.2 Wegweiser durch die Dokumentation zu openUTM 14
1.2.1 openUTM-Dokumentation 14
1.2.2 Dokumentation zum openSEAS-Produktumfeld 18
1.2.3 Readme-Dateien 19

1.3 Neuerungen in openUTM V6.0 20
1.3.1 UTM-Cluster-Anwendung 20
1.3.2 Neue Plattformen 23
1.3.3 Neue Server-Funktionen 23
1.3.4 Neue Client-Funktionen 25
1.3.5 Erweiterungen für openUTM WinAdmin 26

1.4 Darstellungsmittel 27

2 Einführung in die Generierung 29

2.1 UTM-Anwendung konfigurieren 31

2.2 Anwendungskomponenten generieren - Ergebnis des KDCDEF-Laufs 33

2.3 Die KDCFILE 44
2.3.1 Verwaltungsdaten 48
2.3.2 Pagepool 48
2.3.3 Wiederanlaufbereich 51
2.3.4 Neue KDCFILE im laufenden Betrieb erzeugen 54

2.4 Performance-Tuning 56
2.4.1 KDCFILE aufspalten 56
2.4.2 KDCFILE auf raw-device (Unix-Systeme) 58
2.4.3 KDCFILE auf Stripe Set (Windows-Systeme) 61

3 Hinweise zur Generierung einer UTM-Cluster-Anwendung . . . 63

3.1 Generierung einer UTM-Cluster-Anwendung . . . 64
3.2 Generierung einer Reserve-Knoten-Anwendung . . . 68
3.3 Nutzung globaler Speicherbereiche . . . 69
3.4 Verwendung von Benutzern mit RESTART=YES . . . 69
3.5 Besonderheiten in BS2000/OSD . . . 70
3.6 Besonderheiten auf Unix-/Windows-Systemen . . . 70
3.7 Besonderheiten bei LU6.1-Verbindungen . . . 70

4 Anwendungen für verteilte Verarbeitung generieren . . . 71

4.1 Verteilte Verarbeitung über das LU6.1-Protokoll . . . 72
4.1.1 Transportverbindungen und SNA-Sessions . . . 72
4.1.2 Generierungshinweise . . . 74
4.1.3 Vorgehensweise bei der Generierung von LU6.1-Verbindungen . . . 76
4.1.4 LU6.1-LPAP-Bündel . . . 83
4.1.5 LU6.1-LPAP-Bündel einer stand-alone Anwendung mit einer UTM-Cluster-Anwendung . . . 85

4.2 Verteilte Verarbeitung über das OSI TP-Protokoll . . . 88
4.2.1 OSI-Begriffe . . . 88
4.2.2 Generierung für die verteilte Verarbeitung über OSI TP . . . 94
4.2.3 OSI-LPAP-Bündel . . . 100

4.3 UTM- und BCAM-Generierung abstimmen (BS2000/OSD) . . . 104

4.4 Adressinformationen für das Transportsystem CMX bereitstellen (Unix- und Windows-Systeme) . . . 105
4.4.1 Adressinformationen mit KDCDEF bereitstellen . . . 106
4.4.2 Adressinformationen mit TNS-Einträgen bereitstellen . . . 109
4.4.3 Umstellung der Adressinformation von TNS-Einträgen auf KDCDEF . . . 110

4.5 Adressinformationen für das Transportsystem SOCKET bereitstellen (Unix- und Windows-Systeme) . . . 114

4.6 Multi- und single-threaded Netzanbindung (Unix- und Windows-Systeme) . . . 116

4.7 Mapped Hostnamen verwenden (Unix- und Windows-Systeme) . . . 118
4.7.1 Umwandlungsdatei für Mapped Hostnamen . . . 119
4.7.2 Umgebungsvariable UTM_NET_HOSTNAME . . . 121

5 Hinweise zur Generierung ausgewählter Objekte und Funktionen der Anwendung . . . 123

5.1 Clients an die Anwendung anschließen . . . 124
5.1.1 Clients über LTERM-Partner anschließen . . . 125
5.1.2 LTERM-Pools . . . 129
5.1.3 LTERM-Bündel . . . 133
5.1.4 LTERM-Gruppen . . . 136
5.1.5 OpenCPIC-Clients anschließen . . . 139
5.1.6 Anmeldeverfahren für Clients definieren . . . 140
5.1.6.1 Automatischer Verbindungsaufbau . . . 141
5.1.6.2 Automatisches Anmelden unter einer bestimmten Benutzerkennung . . . 141
5.1.6.3 Anmelde-Vorgänge für Clients generieren . . . 141
5.1.6.4 Mehrfachanmeldung . . . 143
5.1.7 Maximale Wartezeiten für die Dialog-Führung festlegen . . . 144
5.1.8 Securityfunktionen generieren . . . 145
5.1.8.1 Zugangsschutz definieren . . . 145
5.1.8.2 Administrationsberechtigungen vergeben . . . 146
5.1.9 Wiederanlauf generieren . . . 147
5.1.10 USP-Header für Ausgabe-Nachrichten auf Socket-Verbindungen . . . 148
5.1.11 Code-Konvertierung . . . 149
5.1.12 Adressinformationen bereitstellen . . . 150
5.1.12.1 Adressinformationen für Clients vom Typ SOCKET bereitstellen . . . 150
5.1.12.2 Adressinformationen für Clients vom Typ UPIC und APPLI in BS2000/OSD bereitstellen . . . 152
5.1.12.3 Adressinformationen für Clients vom Typ UPIC und APPLI in Unix- und Windows-Systemen bereitstellen . . . 153
5.1.12.4 Besonderheiten bei LTERM-Pools auf Unix- und Windows-Systemen . . . 156
5.1.12.5 Adressinformationen für OpenCPIC-Clients in BS2000/OSD bereitstellen . . . 157
5.1.12.6 Adressinformationen für OpenCPIC-Clients in Unix- und Windows-Systemen bereitstellen . . . 158
5.1.13 Beispiele für die Generierung eines Client/Server-Verbundes . . . 160

5.2 Drucker generieren (auf BS2000/OSD und Unix-Systemen) . . . 165
5.2.1 RSO-Drucker generieren (BS2000/OSD) . . . 169
5.2.1.1 Einträge für die KDCDEF-Generierung . . . 169
5.2.1.2 Einträge bei RSO und SPOOL . . . 170
5.2.1.3 Drucker für openUTM aktivieren . . . 174
5.2.1.4 Druckerinformationen abfragen . . . 174
5.2.1.5 Drucker freigeben im Fehlerfall . . . 175
5.2.2 Druckerbündel generieren . . . 175
5.2.3 Bypass-Betrieb (BS2000/OSD) . . . 175
5.2.4 Druckersteuer-LTERMs generieren . . . 176

5.3 Service-gesteuerte Queues generieren . . . 178
5.3.1 USER-Queues . . . 178
5.3.2 TAC-Queues . . . 179
5.3.3 Temporäre Queues . . . 181
5.3.4 Maximale Wartezeit beim Lesen aus Service-gesteuerten Queues festlegen . . . 182
5.3.5 Maximale Anzahl der erneuten Zustellungen an Service-gesteuerte Queues begrenzen . . . 182

5.4 UTM-Meldungen . . . 183
5.4.1 Meldungen in openUTM auf BS2000/OSD . . . 183
5.4.2 Meldungen in openUTM auf Unix- und Windows-Systemen . . . 186
5.4.3 Benutzer-spezifische Meldungsziele . . . 188

5.5 Nachrichtenverteilung und Multiplexing mit OMNIS (BS2000/OSD) . . . 189
5.5.1 Multiplexverbindungen . . . 190
5.5.1.1 Multiplexverbindungen definieren . . . 191
5.5.1.2 Asynchron-Nachrichten ohne Vorankündigung ausgeben . . . 193
5.5.1.3 Bestätigung des Verbindungsabbaus durch den Partner . . . 193
5.5.2 Statistiken zu Multiplexverbindungen . . . 194
5.5.3 Kombination von Multiplexverbindungen und direkten Verbindungen . . . 194

5.6 Lademodule, Common Memory Pools und Shared Code generieren (BS2000/OSD) . . . 196
5.6.1 Lademodule generieren . . . 196
5.6.2 Shared Code und Common Memory Pools generieren . . . 201
5.6.2.1 Shared Code im Systemspeicher . . . 201
5.6.2.2 Shared Code im Common Memory Pool . . . 202

5.7 Auftragssteuerung - Prioritäten und Prozessbeschränkung . . . 205
5.7.1 Auftragsbearbeitung über Prioritäten steuern . . . 208
5.7.2 Auftragsbearbeitung durch Prozessbeschränkung für TAC-Klassen steuern . . . 211
5.7.3 Gegenüberstellung einiger Eigenschaften der beiden Verfahren . . . 212
5.7.4 Prozessprioritäten in BS2000/OSD . . . 215

5.8 Zugriffskontrolle . . . 216
5.8.1 Lock-/Keycode-Konzept . . . 216
5.8.2 Access-List-Konzept . . . 219
5.8.3 Zugriffskontrolle bei verteilter Verarbeitung . . . 223

5.9 Nachrichten-Verschlüsselung auf Verbindungen zu Clients . . . 225
5.9.1 Voraussetzungen . . . 225
5.9.2 Verschlüsselungs-Verfahren . . . 226
5.9.3 Verschlüsselung von Passwörtern und Benutzerdaten . . . 227
5.9.3.1 Zugangsschutz . . . 228
5.9.3.2 Zugriffsschutz . . . 229
5.9.4 RSA-Schlüsselpaar erzeugen und öffentlichen Schlüssel auslesen . . . 231

5.10 Datenbankkopplung definieren . . . 232
5.10.1 Datenbankkopplung unter BS2000/OSD . . . 232
5.10.2 Kopplung mit einem Resource Manager unter Unix- und Windows-Systemen . . . 233

5.11 Internationalisierung der Anwendung - XHCS Unterstützung (BS2000/OSD) . . 235
5.11.1 XHCS-Begriffsdefinitionen . . . 236
5.11.2 Sprachumgebung festlegen - Locale definieren . . . 238
5.11.3 Zeichensatznamen für Editprofile und Formate . . . 241

5.12 Zugangskontrolle mit Kerberos generieren (BS2000/OSD) . . . 242

6 Generierungstool KDCDEF . . . 243

6.1 ROOT-Tabellen-Source und KDCFILE erstellen . . . 243
6.1.1 Anweisungen zur Steuerung des KDCDEF-Laufs . . . 244
6.1.2 Anweisungen zum Erzeugen der ROOT-Tabellen-Source . . . 244
6.1.3 Basisanweisungen zum Erzeugen einer KDCFILE . . . 245
6.1.3.1 KDCFILE erzeugen - zusätzliche Anweisungen für verteilte Verarbeitung über LU6.1 . . . 246
6.1.3.2 KDCFILE erzeugen - zusätzliche Anweisungen für verteilte Verarbeitung über OSI TP . . . 247
6.1.3.3 KDCFILE erzeugen - Anweisungen zum Erzeugen einer Cluster-Konfigurationsdatei . . . 247
6.1.4 Auswirkungen der KDCDEF-Anweisungen auf die Generierungsobjekte . . . 248

6.2 Aufruf von KDCDEF und Eingabe der Steueranweisungen . . . 251
6.2.1 KDCDEF starten und KDCDEF-Lauf durchführen . . . 251
6.2.1.1 BS2000/OSD . . . 251
6.2.1.2 Unix-Systeme . . . 252
6.2.1.3 Windows-Systeme . . . 253
6.2.2 Reihenfolge der Steueranweisungen . . . 254
6.2.3 Format der Steueranweisungen . . . 255
6.2.4 Fortsetzungszeilen in Steueranweisungen . . . 255
6.2.5 Syntax- und Plausibilitätsprüfung . . . 256
6.2.6 KDCDEF-Protokollierung . . . 256
6.2.7 Format und Eindeutigkeit der Objektnamen . . . 257
6.2.7.1 Reservierte Namen . . . 257
6.2.7.2 Format der Namen . . . 257
6.2.7.3 Anzahl der Namen . . . 259
6.2.7.4 Eindeutigkeit der Namen und Adressen . . . 262
6.2.8 Ergebnis des KDCDEF-Laufs . . . 264

6.3 Inverser KDCDEF 265
6.3.1 Inversen KDCDEF starten 267
6.3.2 Ergebnis des inversen KDCDEF 268
6.3.3 Erzeugen von KDCDEF-Steueranweisungen bei Versionsübergängen 270

6.4 Empfehlungen für die Neugenerierung einer Anwendung 270

6.5 Steueranweisungen von KDCDEF 273
ABSTRACT-SYNTAX - Abstrakte Syntax definieren 273
ACCESS-POINT - OSI TP-Zugriffspunkt einrichten 275
ACCOUNT - Accounting-Funktionen festlegen 281
APPLICATION-CONTEXT - Application Context definieren 283
AREA - Zusätzliche Datenbereiche (Areas) definieren 286
BCAMAPPL - Weitere Anwendungsnamen definieren 289
CLUSTER – Globale Eigenschaften einer UTM-Cluster-Anwendung definieren . . . 297
CLUSTER-NODE – Knoten-Anwendung einer UTM-Cluster-Anwendung definieren . 306
CON - Verbindung für die verteilte Verarbeitung über LU6.1 definieren 309
CREATE-CONTROL-STATEMENTS - KDCDEF-Steueranweisungen erzeugen . . . 314
DATABASE - Datenbanksystem definieren (BS2000/OSD) 317
DEFAULT - Standardwerte definieren (BS2000/OSD) 320
EDIT - Editoptionen definieren (BS2000/OSD) 324
EJECT - Seitenvorschub im Protokoll veranlassen 328
END - KDCDEF-Eingabe beenden 329
EXIT - Event-Exits definieren 330
FORMSYS - Formatierungssystem beschreiben 332
KSET - Keyset definieren 334
LOAD-MODULE - Lademodule (BLS) beschreiben (BS2000/OSD) 336
LPAP - LPAP-Partner für die verteilte Verarbeitung über LU6.1 definieren 341
LSES - Sessionnamen für die verteilte Verarbeitung über LU6.1 definieren 345
LTAC - Transaktionscode für Partner-Anwendung definieren 347
LTERM - LTERM-Partner für Clients und Drucker definieren 354
MASTER-LU61-LPAP – Master-LPAP eines LU6.1-LPAP-Bündels definieren 366
MASTER-OSI-LPAP - Master-LPAP eines OSI-LPAP-Bündels definieren 367
MAX - UTM-Anwendungsparameter definieren 368
MESSAGE - Meldungsmodul beschreiben 407
MPOOL - Common Memory Pool beschreiben (BS2000/OSD) 411
MSG-DEST - Benutzer-spezifische Meldungsziele definieren 414
MUX - Multiplexanschluss definieren (BS2000/OSD) 417
OPTION - KDCDEF-Lauf steuern 419
OSI-CON - Logische Verbindungen zum OSI TP-Partner definieren 423
OSI-LPAP - OSI-LPAP-Partner für verteilte Verarbeitung über OSI TP definieren . . 429
PROGRAM - Teilprogramme definieren 437
PTERM - Eigenschaften von Clients und Druckern und Zuordnung zum LTERM-Partner festlegen 440
QUEUE - Tabelleneinträge für Temporäre Queues reservieren 461

REMARK - Kommentarzeile einfügen . . . 463
RESERVE - Tabellenplätze für UTM-Objekte reservieren . . . 464
RMXA - Namen für XA-(Datenbank-)Anschluss definieren (Unix-, Windows-Systeme) . . . 469
ROOT - Namen für ROOT-Tabellen-Source vergeben . . . 470
SATSEL - SAT-Protokollierung steuern (BS2000/OSD) . . . 471
SESCHA - Session-Eigenschaften für verteilte Verarbeitung über LU6.1 festlegen . . . 474
SFUNC - Funktionstasten definieren . . . 477
SHARED-OBJECT - Shared Objects/DLLs definieren (Unix-, Windows-Systeme) . . . 480
SIGNON - Anmeldeverfahren steuern . . . 483
TAC - Eigenschaften von Transaktionscodes und TAC-Queues festlegen . . . 488
TACCLASS - Prozess-Anzahl für TAC-Klassen festlegen . . . 506
TAC-PRIORITIES - Prioritäten der TAC-Klassen festlegen . . . 512
TCBENTRY - Gruppe von TCB-Entries definieren (BS2000/OSD) . . . 516
TLS - Namen von TLS-Blöcken festlegen . . . 517
TPOOL - LTERM-Pools definieren . . . 518
TRANSFER-SYNTAX - Transfersyntax definieren . . . 536
ULS - Namen von ULS festlegen . . . 537
USER - Benutzerkennungen definieren . . . 538
UTMD - Anwendungsparameter für verteilte Verarbeitung festlegen . . . 553

6.6 Dialog-Führung - Einfluss von Generierungsparametern . . . 557

6.7 Beispielgenerierung *Comfo*TRAVEL . . . 559
6.7.1 KDCDEF-Inputdatei DYNAMIC.RMS für UTM-D-Anwendung RMS . . . 560
6.7.2 KDCDEF-Anweisungen für UTM-D-Anwendung RMS . . . 563
6.7.3 KDCDEF-Inputdatei DynamicTravel für UTM-Anwendung TRAVEL . . . 569
6.7.4 KDCDEF-Anweisungen für die UTM-Anwendung TRAVEL . . . 571

6.8 Meldungen von KDCDEF . . . 576

7 Konfiguration einer Anwendung dynamisch ändern . . . 577

7.1 Plätze in den Objekttabellen der KDCFILE reservieren . . . 579

7.2 Voraussetzungen für das dynamische Eintragen von Objekten . . . 581

8 Das Tool KDCUPD - KDCFILE aktualisieren . . . 585

8.1 Arbeitsschritte zum Aktualisieren der KDCFILE für stand-alone UTM-Anwendungen . . . 590

8.2 Arbeitsschritte zum Aktualisieren der KDCFILE für UTM-Cluster-Anwendungen 595

8.3 Steueranweisungen für KDCUPD 597
CATID - Catid für die alte und neue KDCFILE angeben 598
CHECK - KDCFILE auf Konsistenz überprüfen 598
END - Eingabe beenden und Verarbeitung starten 599
KDCFILE - Basisname der alten und neuen KDCFILE angeben 599
LIST - Ablaufprotokoll steuern 600
TRANSFER - Übertragung der Benutzerdaten steuern 602

8.4 Ablaufprotokoll und Meldungen von KDCUPD 606

9 Anhang 609

9.1 Code-Konvertierungstabellen in BS2000/OSD 609

9.2 Code-Konvertierungstabellen in Unix- und Windows-Systemen 610
9.2.1 Code-Tabelle in Unix-Systemen modifizieren 610
9.2.2 Code-Tabelle in Windows-Systemen modifizieren 611

Fachwörter 613

Abkürzungen 653

Literatur 659

Stichwörter 667

1 Einleitung

Moderne unternehmensweite IT-Umgebungen unterliegen zahlreichen Herausforderungen von zunehmender Brisanz. Dies wird verursacht durch

- heterogene Systemlandschaften
- unterschiedliche HW-Plattformen
- unterschiedliche Netze und Netzzugriffe (TCP/IP, SNA, HTTP)
- Verflechtung der Anwendungen mit den Unternehmen

Dadurch entwickeln sich Problemfelder, sei es bei Fusionen, durch Kooperationen oder auch nur durch Rationalisierungsmaßnahmen. Die Unternehmen fordern flexible und skalierbare Anwendungen, gleichzeitig soll die Transaktionssicherheit für Prozesse und Daten gewährleistet bleiben, obwohl die Geschäftsprozesse immer komplexer werden. Die wachsende Globalisierung geht selbstverständlich davon aus, dass Anwendungen im 7x24-Stunden-Betrieb laufen und hochverfügbar sind, um beispielsweise Internetzugriffe auf bestehende Anwendungen über Zeitzonen hinweg zu ermöglichen.

Die transaktionsorientierte Middleware-Plattform openUTM bietet eine Ablaufumgebung, die allen diesen Anforderungen moderner unternehmenskritischer Anwendungen gewachsen ist, denn openUTM verbindet alle Standards und Vorteile von transaktionsorientierten Middleware-Plattformen und Message Queuing Systemen:

- Konsistenz der Daten und der Verarbeitung
- Hohe Verfügbarkeit der Anwendungen (nicht nur der Hardware)
- Hohen Durchsatz auch bei großen Benutzerzahlen, d.h. höchste Skalierbarkeit
- Flexibilität bezüglich Änderungen und Anpassungen des IT-Systems

openUTM kann auf einem einzelnen Rechner als stand-alone UTM-Anwendung und auf mehreren Rechnern als UTM-Cluster-Anwendung betrieben werden.

openUTM ist Teil des umfassenden Angebots von **openSEAS**. Gemeinsam mit der Oracle Fusion Middleware bietet openSEAS die komplette Funktionalität für Anwendungsinnovation und moderne Anwendungsentwicklung. Im Rahmen des Produktangebots **openSEAS** nutzen innovative Produkte die ausgereifte Technologie von openUTM:

- BeanConnect ist ein Adapter gemäß der Java Connector Architecture (JCA) von Sun und bietet den standardisierten Anschluss von UTM-Anwendungen an J2EE Application Server. Dadurch können bewährte Legacy-Anwendungen in neue Geschäftsprozesse integriert werden.
- Mit WebTransactions steht in openSEAS ein Produkt zur Verfügung, welches es ermöglicht, bewährte Host-Anwendungen flexibel in neuen Geschäftsprozessen und modernen Einsatzszenarien zu nutzen. Bestehende UTM-Anwendungen können unverändert ins Web übernommen werden.

1.1 Zielgruppe und Konzept des Handbuchs

Das openUTM-Handbuch „Anwendungen generieren“ richtet sich an Anwendungsplaner, Anwendungsentwickler und Betreuer von UTM-Anwendungen.

Dieses Handbuch beschreibt, wie Sie für eine UTM-Anwendung mit Hilfe des UTM-Tools KDCDEF die Konfiguration definieren und die KDCFILE erzeugen. Zusätzlich wird in Kapitel 5 näher auf die Generierung ausgewählter Objekte und Funktionen der Anwendung eingegangen.

Weitere Themen sind die dynamische Konfiguration einer Anwendung und die Aktualisierung der KDCFILE mit dem Tool KDCUPD.

Kenntnisse des Betriebssystems werden vorausgesetzt.

Wenn im Folgenden allgemein von Unix-System bzw. Unix-Plattform die Rede ist, dann ist darunter sowohl ein Unix-basiertes Betriebssystem wie z.B. Solaris oder HP-UX als auch eine Linux-Distribution wie z.B. SUSE oder Red Hat zu verstehen.

Wenn im Folgenden von Windows-System bzw. Windows-Plattform die Rede ist, dann sind damit alle Windows-Varianten gemeint, auf denen openUTM zum Ablauf kommt.

1.2 Wegweiser durch die Dokumentation zu openUTM

In diesem Abschnitt erhalten Sie einen Überblick über die Handbücher zu openUTM und zum Produktumfeld von openUTM.

1.2.1 openUTM-Dokumentation

Die openUTM-Dokumentation besteht aus Handbüchern, einer Online-Hilfe für den grafischen Administrationsarbeitsplatz openUTM WinAdmin sowie einer Freigabemitteilung für jede Plattform, auf der openUTM freigegeben wird.

Es gibt Handbücher, die für alle Plattformen gültig sind, sowie Handbücher, die jeweils für BS2000/OSD bzw. für Unix-Systeme und Windows-Systeme gelten.

Sämtliche Handbücher sind als PDF-Datei im Internet verfügbar unter der Adresse

http://manuals.ts.fujitsu.com

Geben Sie dort in das Feld **Produktsuche** den Suchbegriff "openUTM V6.0" ein, um alle openUTM-Handbücher der Version 6.0 anzeigen zu lassen.

Die Handbücher sind auf offenen Plattformen auf der Enterprise DVD enthalten und stehen außerdem auch auf der WinAdmin DVD (für BS2000/OSD) zur Verfügung.

Die folgenden Abschnitte geben einen Aufgaben-bezogenen Überblick über die Dokumentation zu openUTM V6.0. Eine vollständige Liste der Dokumentation zu openUTM finden Sie im Literaturverzeichnis auf Seite 659.

Einführung und Überblick

Das Handbuch **Konzepte und Funktionen** gibt einen zusammenhängenden Überblick über die wesentlichen Funktionen, Leistungen und Einsatzmöglichkeiten von openUTM. Es enthält alle Informationen, die Sie zum Planen des UTM-Einsatzes und zum Design einer UTM-Anwendung benötigen. Sie erfahren, was openUTM ist, wie man mit openUTM arbeitet und wie openUTM in die BS2000/OSD-, Unix- und Windows-Plattformen eingebettet ist.

Programmieren

- Zum Erstellen von Server-Anwendungen über die KDCS-Schnittstelle benötigen Sie das Handbuch **Anwendungen programmieren mit KDCS für COBOL, C und C++**, in dem die KDCS-Schnittstelle in der für COBOL, C und C++ gültigen Form beschrieben ist. Diese Schnittstelle umfasst sowohl die Basisfunktionen des universellen Transaktionsmonitors als auch die Aufrufe für verteilte Verarbeitung. Es wird auch die Zusammenarbeit mit Datenbanken beschrieben.
- Wollen Sie die X/Open-Schnittstellen nutzen, benötigen Sie das Handbuch **Anwendungen erstellen mit X/Open-Schnittstellen**. Es enthält die UTM-spezifischen Ergänzungen zu den X/Open-Programmschnittstellen TX, CPI-C und XATMI sowie Hinweise zu Konfiguration und Betrieb von UTM-Anwendungen, die X/Open-Schnittstellen nutzen. Ergänzend dazu benötigen Sie die X/Open-CAE-Specification für die jeweilige X/Open-Schnittstelle.
- Wenn Sie Daten auf Basis von XML austauschen wollen, benötigen Sie das Dokument **XML für openUTM**. Darin werden die C- und COBOL-Aufrufe beschrieben, die zum Bearbeiten von XML-Dokumenten benötigt werden. Diese Beschreibung ist online als PDF-Dokument erhältlich.
- Für BS2000/OSD gibt es Ergänzungsbände für die Programmiersprachen Assembler, Fortran, Pascal-XT und PL/1. Diese sind online als PDF-Datei verfügbar.

Konfigurieren

Zur Definition von Konfigurationen steht Ihnen das Handbuch **Anwendungen generieren** zur Verfügung. Darin ist beschrieben, wie Sie für eine UTM-Anwendung mit Hilfe des UTM-Tools KDCDEF die Konfiguration definieren und die KDCFILE erzeugen. Außerdem wird die Generierung einer UTM-Cluster-Anwendung beschrieben. Zusätzlich wird gezeigt, wie Sie wichtige Verwaltungs- und Benutzerdaten mit Hilfe des Tools KDCUPD in eine neue KDCFILE übertragen, z.B. beim Umstieg auf eine neue Version von openUTM oder nach Änderungen in der Konfiguration.

Binden, Starten und Einsetzen

Um UTM-Anwendungen einsetzen zu können, benötigen Sie für das betreffende Betriebssystem (BS2000/OSD bzw. Unix-/Windows-Systeme) das Handbuch **Einsatz von openUTM-Anwendungen**. Dort ist beschrieben, wie man ein UTM-Anwendungsprogramm bindet und startet, wie man sich bei einer UTM-Anwendung an- und abmeldet und wie man Anwendungsprogramme strukturiert und im laufenden Betrieb austauscht. Außerdem enthält es die UTM-Kommandos, die dem Terminal-Benutzer zur Verfügung stehen. Zudem wird ausführlich auf die Punkte eingegangen, die beim Betrieb von UTM-Cluster-Anwendungen zu beachten sind.

Administrieren und Konfiguration dynamisch ändern

- Für das Administrieren von Anwendungen finden Sie die Beschreibung der Programmschnittstelle zur Administration und die UTM-Administrationskommandos im Handbuch **Anwendungen administrieren**. Es informiert über die Erstellung eigener Administrationsprogramme für den Betrieb einer stand-alone UTM-Anwendung und einer UTM-Cluster-Anwendung sowie über die Möglichkeiten, mehrere UTM-Anwendungen zentral zu administrieren. Darüber hinaus beschreibt es, wie Sie Message Queues und Drucker mit Hilfe der KDCS-Aufrufe DADM und PADM administrieren können.
- Wenn Sie den grafischen Administrationsarbeitsplatz **openUTM WinAdmin** einsetzen, dann steht Ihnen folgende Dokumentation zur Verfügung:
  - Die **WinAdmin-Beschreibung** bietet einen umfassenden Überblick über den Funktionsumfang von WinAdmin und das Handling von WinAdmin. Dieses Dokument wird mit der Software ausgeliefert und ist zusätzlich auch online als PDF-Datei verfügbar.
  - Das **Online-Hilfesystem** beschreibt kontextsensitiv alle Dialogfelder und die zugehörigen Parameter, die die grafische Oberfläche bietet. Außerdem wird dargestellt, wie man WinAdmin konfiguriert, um UTM-Anwendungen administrieren und generieren zu können.

Testen und Fehler diagnostizieren

Für die o.g. Aufgaben benötigen Sie außerdem die Handbücher **Meldungen, Test und Diagnose** (jeweils ein Handbuch für Unix-/Windows-Systeme und für BS2000/OSD). Sie beschreiben das Testen einer UTM-Anwendung, den Inhalt und die Auswertung eines UTM-Dumps, das Verhalten im Fehlerfall, das Meldungswesen von openUTM, sowie alle von openUTM ausgegebenen Meldungen und Returncodes.

openUTM-Clients erstellen

Wenn Sie Client-Anwendungen für die Kommunikation mit UTM-Anwendungen erstellen wollen, stehen Ihnen folgende Handbücher zur Verfügung:

- Das Handbuch **openUTM-Client für Trägersystem UPIC** beschreibt Erstellung und Einsatz von Client-Anwendungen, die auf UPIC basieren. Neben der Beschreibung der Schnittstellen CPI-C und XATMI erhalten Sie Informationen, wie Sie die C++-Klassen oder ActiveX für die schnelle und einfache Programmerstellung nutzen können.
- Das Handbuch **openUTM-Client für Trägersystem OpenCPIC** beschreibt, wie man OpenCPIC installiert und konfiguriert. Es zeigt auf, was beim Programmieren einer CPI-C-Anwendung zu beachten ist und welche Einschränkungen es gegenüber der Programmschnittstelle X/Open CPI-C gibt.
- Für die mit **BeanConnect** ausgelieferten **JUpic-Java-Klassen** wird die Dokumentation mit der Software ausgeliefert. Diese Dokumentation besteht aus Word- und PDF-Dateien, die die Einführung und die Installation beschreiben, sowie aus einer Java-Dokumentation mit der Beschreibung der Java-Klassen.
- Das Handbuch **BizXML2Cobol** beschreibt, wie Sie bestehende Cobol-Programme einer UTM-Anwendung so erweitern können, dass sie als Standard-Web-Service auf XML-Basis genutzt werden können. Die Arbeit mit der grafischen Bedienoberfläche ist in der zugehörigen **Online-Hilfe** beschrieben.
- Wenn Sie UTM-Services auf einfache Weise ins Web stellen möchten, benötigen Sie das Handbuch **Web-Services für openUTM**. Das Handbuch beschreibt, wie Sie mit dem Software-Produkt WS4UTM (WebServices for openUTM) Services von UTM-Anwendungen als Web Services verfügbar machen. Die Arbeit mit der grafischen Bedienoberfläche ist in der zugehörigen **Online-Hilfe** beschrieben.

Kopplung mit der IBM-Welt

Wenn Sie aus Ihrer UTM-Anwendung mit Transaktionssystemen von IBM kommunizieren wollen, benötigen Sie außerdem das Handbuch **Verteilte Transaktionsverarbeitung zwischen openUTM und CICS-, IMS- und LU6.2-Anwendungen**. Es beschreibt die CICS-Kommandos, IMS-Makros und UTM-Aufrufe, die für die Kopplung von UTM-Anwendungen mit CICS- und IMS-Anwendungen benötigt werden. Die Kopplungsmöglichkeiten werden anhand ausführlicher Konfigurations- und Generierungsbeispiele erläutert. Außerdem beschreibt es die Kommunikation über openUTM-LU62, sowie dessen Installation, Generierung und Administration.

1.2.2 Dokumentation zum openSEAS-Produktumfeld

Die Verbindung von openUTM zum openSEAS-Produktumfeld wird im openUTM-Handbuch **Konzepte und Funktionen** kurz dargestellt. Die folgenden Abschnitte zeigen, welche der openSEAS-Dokumentationen für openUTM von Bedeutung sind.

Integration von J2EE Application Servern und UTM-Anwendungen

Der Adapter BeanConnect gehört zur Produkt-Suite openSEAS. Der BeanConnect-Adapter realisiert die Verknüpfung zwischen klassischen Transaktionsmonitoren und J2EE Application Servern und ermöglicht damit die effiziente Integration von Legacy-Anwendungen in Java-Anwendungen.

- Das Handbuch **BeanConnect** beschreibt das Produkt BeanConnect, das einen JCA 1.5-konformen Adapter bietet, der UTM-Anwendungen mit Anwendungen auf Basis von J2EE, z.B. mit dem Application Server von Oracle, verbindet.

  Die Handbücher zum Application Server von Oracle sind bei Oracle beziehbar.

Web-Anbindung und Anwendungsintegration

Zum Anschließen neuer und bestehender UTM-Anwendungen an das Web mit dem Produkt WebTransactions benötigen Sie die Handbücher zu WebTransactions.

Die Dokumentation wird durch JavaDocs ergänzt.

1.2.3 Readme-Dateien

Funktionelle Änderungen und Nachträge der aktuellen Produktversion zu diesem Handbuch entnehmen Sie bitte ggf. den Produkt-spezifischen Readme-Dateien.

- BS2000/OSD:

  Sie finden die Informationen auf Ihrem BS2000-Rechner in der Freigabemitteilung (Dateiname SYSFGM.UTM.060.D*)* und eventuell zusätzlich in einer Readme-Datei (Dateiname SYSRME.UTM.060.D*)*. Die Benutzerkennung, unter der sich die Datei befindet, erfragen Sie bitte bei Ihrer zuständigen Systembetreuung. Die Datei können Sie mit dem Kommando /SHOW-FILE oder mit einem Editor ansehen oder auf einem Standarddrucker mit folgendem Kommando ausdrucken:

  `/PRINT-DOCUMENT` *dateiname*`,LINE-SPACING=*BY-EBCDIC-CONTROL`

- Unix-Systeme:

  Die Readme-Datei und ggf. weitere Dateien wie z.B. eine Handbuchergänzungsdatei finden Sie im *utmpfad* unter `/docs/`*sprache*.

- Windows-Systeme:

  Die Readme-Datei und ggf. weitere Dateien wie z.B. eine Handbuchergänzungsdatei finden Sie im *utmpfad* unter `\Docs\`*sprache*.

1.3 Neuerungen in openUTM V6.0

openUTM ermöglicht es, UTM-Anwendungen in einer Mehrrechner-Konfiguration (Cluster) zu einer UTM-Cluster-Anwendung zusammenzufassen. Außerdem bietet openUTM eine Reihe neuer Funktionen auf Server- und auf Client-Seite. Der grafische Administrationsarbeitsplatz WinAdmin wurde ebenfalls funktionell erweitert.

Die folgenden Abschnitte gehen näher auf die Neuerungen in den einzelnen Bereichen ein.

1.3.1 UTM-Cluster-Anwendung

Eine UTM-Anwendung kann so generiert werden, dass sie sich in einer Mehrrechner-Konfiguration (Cluster) in identischer Form auf die Knoten des Clusters verteilen und als UTM-Cluster-Anwendung betreiben lässt. Eine UTM-Cluster-Anwendung bietet im Vergleich zu einer stand-alone UTM-Anwendung zusätzliche Möglichkeiten:

- Echten 7x24-Stunden-Betrieb der UTM-Cluster-Anwendung durch Höchstverfügbarkeit infolge von Online-Update, Online-Import und Anwendungsüberwachung.

  Die Höchstverfügbarkeit wird durch Einsatz mehrerer Knoten-Anwendungen mit gegenseitiger Überwachung der Knoten-Anwendungen erreicht. Eine UTM-Cluster-Anwendung kann in BS2000/OSD auf bis zu 16 Knoten und in Unix- oder Windows-Systemen auf bis zu 32 Knoten verteilt werden.

- Globale Administration aller Knoten-Anwendungen.

Zentrale Ablage der Cluster-Konfiguration (single system image)

Für den Betrieb eines Clusters müssen Sie alle Cluster-globalen Dateien so ablegen, dass alle Knoten-Anwendungen auf sie Zugriff haben. Dies sind u.a. folgende Dateien:

- Die zentrale Cluster-Konfigurationsdatei, die die Konfiguration des Clusters enthält. Diese Datei wird durch das Dienstprogramm KDCDEF erzeugt. Beim Starten und beim Beenden einer Knoten-Anwendung werden in dieser Datei die Laufzeitinformationen zu den Knoten-Anwendungen aktualisiert.

- Die zentrale Cluster-User-Datei, die zur Verwaltung der Benutzer einer UTM-Cluster-Anwendung dient. Diese Datei kann während des Betriebs einer Anwendung modifiziert werden, z.B. wenn der Administrator neue Benutzer für die UTM-Cluster-Anwendung definiert.

- Das zentrale Administrations-Journal, das zum Nachvollziehen der Cluster-globalen administrativen Änderungen auf allen Knoten-Anwendungen dient. Die Dateien des Administrations-Journals werden im laufenden Betrieb angelegt und gegebenenfalls vergrößert.

- Die KDCFILE-Datei(en) der einzelnen Knoten-Anwendungen. Auf diese muss für die Anwendungsüberwachung und den Online-Import von allen Knoten aus zugegriffen werden können.

In BS2000/OSD werden die Cluster-globalen Dateien auf einem Shared-Pubset-Verbund (XCS) abgelegt, auf Unix-Systemen auf einem NFS-Netzlaufwerk und auf Windows-Systemen auf Netzlaufwerken. Der Zugriff der einzelnen Knoten-Anwendungen auf die globalen Cluster-Dateien ist Rechner-übergreifend serialisiert.

Anwendungsüberwachung

Die Knoten-Anwendungen überwachen sich gegenseitig. Bei Ausfall einer Knoten-Anwendung können Aktionen veranlasst werden. Im Einzelnen heißt das:

- Beim Start einer Knoten-Anwendung wird dynamisch bestimmt, welche andere Knoten-Anwendung von dieser Knoten-Anwendung überwacht werden soll, und welche andere Knoten-Anwendung diese Knoten-Anwendung überwacht.

  Diese Überwachungsbeziehung wird in der Cluster-Konfigurationsdatei eingetragen. Beim Beenden der Knoten-Anwendung wird diese Beziehung wieder aufgelöst. Nachdem eine Überwachungsbeziehung eingerichtet wurde, wird in einem festgelegten, generierbaren Intervall die Existenz der überwachten Knoten-Anwendung geprüft.

- Für den Fall, dass der Ausfall einer Knoten-Anwendung erkannt wird, können Sie über eine Prozedur oder ein Skript weitere Folgeaktionen veranlassen.

  Den Namen der Prozedur bzw. des Skripts, das in dieser Situation gestartet werden soll, können Sie per Generierung vorgeben. Der Inhalt der Prozedur bzw. des Skripts hängt von den Aktionen und der Plattform ab, auf der die Knoten-Anwendungen ablaufen. Der Inhalt ist vom Anwender festzulegen.

Online-Import von Anwendungsdaten

Nachdem eine Knoten-Anwendung normal beendet wurde, können noch nicht ausgeführte Aufträge von anderen, laufenden Knoten-Anwendungen übernommen werden.

Generierung

Für die Generierung einer UTM-Cluster-Anwendung bietet openUTM folgende Generierungsanweisungen:

- CLUSTER
  Mit dieser Anweisung legen Sie die globalen Eigenschaften einer UTM-Cluster-Anwendung fest.
- CLUSTER-NODE
  Mit dieser Anweisung legen Sie die Eigenschaften einer Knoten-Anwendung fest. Sie muss einmal pro Knoten-Anwendung angegeben werden.

Administration

Eine UTM-Cluster-Anwendung kann über die Programmschnittstelle KDCADMI administriert werden. Dazu wurde KDCADMI (siehe unten) entsprechend erweitert. Das Administrations-Tool openUTM WinAdmin verwendet diese Programmschnittstelle und wurde für die Administration von UTM-Cluster-Anwendungen ebenfalls erweitert (siehe Seite 26).

Je nach Administrationsauftrag wirken administrative Änderungen entweder nur lokal in der einzelnen Knoten-Anwendung, an der der Administrator angemeldet ist, oder global in allen Knoten-Anwendungen. **Globale** Änderungen werden in das **Administrations-Journal** geschrieben. Anhand des Administrations-Journals vollziehen alle Knoten-Anwendungen diese Änderungen nach.

KDCADMI

Für die Administration von UTM-Cluster-Anwendungen bietet KDCADMI folgende neuen Operationscodes und Datenstrukturen:

- Datenstruktur *kc_cluster_par_str*:
  Eigenschaften einer UTM-Cluster-Anwendung
- Datenstruktur *kc_cluster_node_str*:
  Eigenschaften einer Knoten-Anwendung
- Datenstrukturen *kc_user_fix_str*, *kc_user_dyn1_str* und *kc_user_dyn2_str*:
  Benutzerkennungen in einer UTM-Cluster-Anwendung. Diese Datenstrukturen können auch in stand-alone UTM-Anwendungen genutzt werden.
- Operationscode KC_LOCK_MGMT mit Datenstruktur *kc_lock_mgmt_str*:
  Sperre in Cluster-User-Datei aufheben
- Operationscode KC_ONLINE_IMPORT mit Datenstruktur *kc_online_import_str*:
  Anwendungsdaten online importieren

Prozeduren/Skripte

Der Betrieb einzelner Knoten-Anwendungen kann über verschiedene Prozeduren bzw. Skripte von außen gesteuert werden.

1.3.2 Neue Plattformen

openUTM ist auf HP-UX (PA-Risc), AIX und Linux Intel/AMD zusätzlich im 64 Bit-Modus verfügbar.

Im Cluster-Betrieb ist openUTM auf BS2000/OSD, Sun Solaris und Linux verfügbar.

Details entnehmen Sie der jeweiligen Freigabemitteilung.

1.3.3 Neue Server-Funktionen

Kommunikation mit Partner-Anwendungen

- LU6.1-LPAP-Bündel

  LU6.1-LPAP-Bündel dienen der Lastverteilung. Ein Bündel besteht aus einem Master-LPAP und mehreren Slave-LPAPs. Die Slave-LPAPs werden dem Master-LPAP bei der Generierung zugeordnet.

  Dialog- und Asynchron-Aufträge an das Master-LPAP eines LU6.1-LPAP-Bündels werden zyklisch auf die aufgebauten parallelen Verbindungen des Bündels verteilt und an die Partner-Anwendungen gesendet.

Datenbank-Anschluss

- Oracle BS2000 ohne IUTMDB

  Im BS2000/OSD wird Oracle ab 10g unterstützt. Oracle 10g unterstützt nur die Anbindung über das XA-Interface, nicht mehr über die IUTMDB-Schnittstelle.

- Prüfung des Oracle DB-Anschluss in BS2000/OSD

  Beim Anwendungsstart wird für Oracle DB-Anwendungen geprüft, ob ein statisch gebundener Verbindungsmodul zum generierten Datenbankanschluss passt.

Meldungen

- Unterstützung längerer Meldungstexte

  Die maximale Länge für geänderte und selbst erstellte Meldungstexte wurde auf 512 Bytes erhöht.

Diagnose

- BTRACE-Erweiterungen

  Sie können wählen, in welcher maximalen Länge die Nachrichten protokolliert werden sollen. Standardwert ist 256 Bytes.

- UTM-DIAGAREA-Aufteilung

  Der UTM-DIAGAREA-Bereich wird aufgeteilt in UTM-DIAGAREA und DB-DIAGAREA. In der UTM-DIAGAREA sind KDCS- und FHS-Aufrufe (BS2000/OSD) protokolliert, die DB-DIAGAREA enthält die DB-Aufrufe.

- ROOTDATA-Bereiche

  Die drei ROOT-Bereiche ROOTDATA_NT, ROOTDATA_TU und ROOTDATA_TF wurden zum Bereich ROOTDATA zusammengefasst.

- KDCDUMP AFIND in Unix-/Windows-Systemen

  Das AFIND-Kommando im Dienstprogramm KDCDUMP wurde erweitert, so dass die Semantik der Eingabe eindeutig ist.

- utmmain-Meldungen in Unix-/Windows-Systemen

  Bei schwerwiegenden Fehlern im utmmain-Prozess, die zu einem abnormalem Anwendungsende führen, wird eine U-Meldung mit der Fehlerursache ausgegeben.

- Windows Event Logging

  Die Installation, Deinstallation und der Betrieb von openUTM-Services auf Windows-Systemen werden in der Ereignisanzeige (Event Viewer) in Form von Ereignissen (Events) protokolliert.

Sonstige Änderungen

- Abmelden UPIC-Benutzer
  Am Ende einer Conversation kann wahlweise auf das Abmelden eines UPIC-Benutzers verzichtet werden. Dies führt inbesondere in einem Cluster zu einem Performance-Gewinn, da es die Anzahl der Zugriffe auf die Cluster-User-Datei minimiert.

- Auslieferung KDCEVAL-Makro in BS2000/OSD

  Das KDCEVAL-Makro wird auf der WinAdmin-DVD und der UTM-DVD ausgeliefert.

- SHOW-ETPND Prozedur im BS2000/OSD

  Die Beispielprozedur SHOW-ETPND verwendet nur noch den Editor EDT.

- Kerberos-Erweiterung in BS2000/OSD

  Beim Verbindungsaufbau von Terminals kann ein Kerberos-Dialog durchgeführt werden, dessen Ergebnis im Teilprogramm gelesen werden kann.

- openSM2-Unterstützung auch in Unix-/Windows-Systemen

  Die betreffenden Daten werden wie im BS2000/OSD aus UTM-Anwendungen ermittelt und von openSM2 ausgewertet.

1.3.4 Neue Client-Funktionen

- Lastverteilung UPIC-Client

  Die Kommunikation von UPIC-Clients zu einer UTM-Cluster-Anwendung kann so konfiguriert werden, dass Aufträge von UPIC-Clients an die UTM-Cluster-Anwendung auf die einzelnen Knoten-Anwendungen verteilt werden. Aus einer Liste von Knoten-Anwendungen wird zufällig eine Knoten-Anwendung ausgewählt, mit der die UPIC-Kommunikation erfolgt.

  Diese Art der Konfiguration können Sie festlegen, indem Sie in der Side Information Datei (upicfile) für jede Knoten-Anwendung einen Eintrag angeben. Um diese einzelnen Einträge zu einer UTM-Cluster-Anwendung zusammenzufassen, verwenden Sie für jeden Eintrag das Cluster-Präfix "CD" und den identischen "symbolic destination name".

1.3.5 Erweiterungen für openUTM WinAdmin

Die folgenden Funktionen wurden in openUTM WinAdmin V6.0 neu eingeführt (Begriffe der WinAdmin-Oberfläche sind in **Fettschrift** gekennzeichnet):

- Unterstützung von UTM-Cluster-Anwendungen.
  Der Kollektionsbaum in WinAdmin wurde erweitert: Die UTM-Cluster-Anwendungen mit ihren einzelnen Knoten-Anwendungen sind unter dem Teilbaum **Cluster** eingeordnet, stand-alone Anwendungen unter dem Teilbaum **Standalone**.
- Unterstützung der Funktion **LU6.1-LPAP-Bündel**
- Unterstützung aller neuen Generierungsparameter
- UTM-Benutzerkennungen:
  Zusätzlich zur Objektliste **Users**, die alle UTM-Benutzerkennungen enthält, bietet der Kollektionsbaum die Liste **Users (expl. gen.)**. Diese Liste enthält alle UTM-Benutzerkennungen, die explizit generiert wurden. Die explizit generierten UTM-Benutzerkennungen sind nach wie vor auch in der Liste **Users** enthalten.
- Kommandomodus:
  Sie können Kommandos einzeln aus der Liste der Kommandos löschen, die im Dialogfeld **Kommandomodus** angezeigt wird.
- BeanConnect:
  - Sie können die Management Console von BeanConnect direkt aus WinAdmin heraus starten, wenn sie ordnungsgemäß auf dem WinAdmin-Rechner installiert ist.
  - Sie können die UTM-Anwendungen der BeanConnect Proxys mit Hilfe der Datei `container.properties` in die Konfigurationsdatenbank von WinAdmin importieren.
- Automatisches Starten beim Anmelden eines Windows-Benutzers:
  WinAdmin kann ohne manuellen Eingriff eines Benutzers gestartet werden. Dazu tragen Sie eine entsprechende Verknüpfung in die Programmgruppe **Autostart** ein. Diese Verknüpfung muss als Kommandozeilenargumente den WinAdmin-Benutzer enthalten, der verwendet werden soll sowie sein Passwort.

1.4 Darstellungsmittel

Metasyntax

Die in diesem Handbuch verwendete Metasyntax können Sie der folgenden Tabelle entnehmen:

Formale Darstellung	Erläuterung	Beispiel
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Konstanten (Namen von Aufrufen, Anweisungen, Feldnamen, Kommandos und Operanden etc.), die in dieser Form anzugeben sind.	`LOAD-MODE=STARTUP`
kleinbuchstaben	In Kleinbuchstaben sind in Syntaxdiagrammen und Operandenbeschreibung die Platzhalter für Operandenwerte dargestellt.	`KDCFILE=filebase`
kleinbuchstaben	Im Fließtext werden Variablen sowie Namen von Datenstrukturen und Feldern in kursiven Kleinbuchstaben dargestellt.	*filebase* ist der Basisname der KDCFILE
`Schreibmaschinen-schrift`	In Schreibmaschinenschrift werden im Fließtext Kommandos, Dateinamen, Meldungen und Beispiele ausgezeichnet, die in genau dieser Form eingegeben werden müssen bzw. die genau diesen Namen oder diese Form besitzen.	Der Aufruf `tpcall`
{ } und |	In geschweiften Klammern stehen alternative Angaben, von denen Sie eine auswählen müssen. Die zur Verfügung stehenden Alternativen werden jeweils durch einen Strich getrennt aufgelistet.	`STATUS={ ON | OFF }`
[ ]	In eckigen Klammern stehen wahlfreie Angaben, die entfallen können.	`KDCFILE=( filebase [, { SINGLE| DOUBLE} ] )`
( )	Kann für einen Operanden eine Liste von Parametern angegeben werden, sind diese in runde Klammern einzuschließen und durch Kommata zu trennen. Wird nur ein Parameter angegeben, kann auf die Klammern verzichtet werden.	`KEYS=(key1,key2,...key`*n*`)`
<u>Unterstreichen</u>	Unterstreichen kennzeichnet den Standardwert.	`CONNECT= { A/YES | NO }`

Formale Darstellung	Erläuterung	Beispiel
Kurzform	Die Standardkurzform für Anweisungen, Operanden und Operandenwerte wird „fett“ hervorgehoben. Die Kurzform kann alternativ angegeben werden.	TRANSPORT-**SEL**ECTOR=c‘C‘
...	Punkte zeigen die Wiederholbarkeit einer syntaktischen Einheit an. Außerdem kennzeichnen die Punkte Ausschnitte aus einem Programm, einer Syntaxbeschreibung o.ä.	KDCDEF starten : : OPTION DATA=statement_file : END

Sonstige Symbole

B
B
BS2000/OSD-spezifische Teile der Beschreibung sind am linken Rand mit dem nebenstehenden Symbol gekennzeichnet.

X
X
Unix-System-spezifische Teile der Beschreibung sind am linken Rand mit dem nebenstehenden Symbol gekennzeichnet.

W
W
Windows-System-spezifische Teile der Beschreibung sind am linken Rand mit dem nebenstehenden Symbol gekennzeichnet.

B/X
B/X
Teile der Beschreibung, die nur für openUTM in BS2000/OSD und Unix-Systemen von Bedeutung sind, sind am linken Rand mit dem nebenstehenden Symbol gekennzeichnet.

B/W
B/W
B/W
Teile der Beschreibung, die nur für openUTM in BS2000/OSD und in Windows-Systemen von Bedeutung sind, sind am linken Rand mit dem nebenstehenden Symbol gekennzeichnet.

X/W
X/W
Teile der Beschreibung, die nur für openUTM in Unix- und Windows-Systemen von Bedeutung sind, sind am linken Rand mit dem nebenstehenden Symbol gekennzeichnet.

für Verweise auf umfassende und detaillierte Informationen zum jeweiligen Thema.

für Hinweistexte.

für Warnhinweise.

2 Einführung in die Generierung

Neben den Teilprogrammen, die die Services erbringen, und Formaten (für den formatierten Betrieb in BS2000/OSD) müssen Sie für eine UTM-Anwendung folgende Anwendungs-Komponenten erstellen:

- Konfigurationsdatei KDCFILE
  Die KDCFILE enthält die Konfiguration Ihrer Anwendung. In der KDCFILE hinterlegt openUTM alle für den Betrieb der Anwendung notwendigen Verwaltungsdaten und reserviert Bereiche für Benutzerdaten und zur Transaktionssicherung. Im Betrieb der Anwendung greifen alle Tasks/Workprozesse der Anwendung auf die KDCFILE zu.
  Wie Sie die KDCFILE erstellen, ist in diesem Handbuch beschrieben.

- Main-Routine KDCROOT
  Unter der Steuerung von KDCROOT laufen die von Ihnen erstellten Teilprogramme ab. Basis der Main-Routine KDCROOT sind die ROOT-Tabellen.
  Die ROOT-Tabellen enthalten anwendungsspezifische Konfigurationsdaten, die von der Main-Routine KDCROOT benötigt werden. Die Main-Routine KDCROOT stellt im Betrieb der Anwendung die Verbindung von openUTM zu den Teilprogrammen, der Datenbank und unter BS2000/OSD dem Formatierungssystem her.
  Wie Sie die Sourcen für die ROOT-Tabellen erstellen, ist in diesem Handbuch beschrieben.

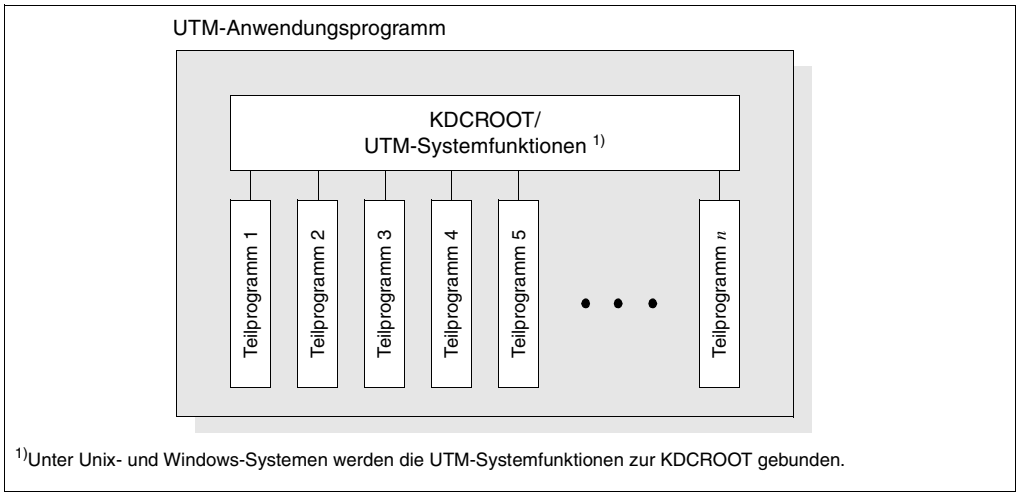


Bild 1: Struktur des UTM-Anwendungsprogramms

Um die Root-Tabellen-Sourcen und die KDCFILE zu erzeugen, müssen Sie zunächst die Konfiguration der UTM-Anwendung definieren. Die hierzu notwendigen Schritte werden unter dem Begriff „Generierung“ zusammengefasst. Für die Definition der Konfiguration und die Generierung der KDCFILE und der ROOT-Tabellen-Source stellt openUTM Ihnen das Generierungstool KDCDEF zur Verfügung.

Das Generierungstool KDCDEF wird im Kapitel „Generierungstool KDCDEF“ ab Seite 243 ausführlich beschrieben.

Informationen zur KDCFILE finden Sie ab Seite 44.

Die Informationen zur Generierung in diesem Kapitel „Einführung in die Generierung“ gelten sowohl für stand-alone UTM-Anwendungen als auch für UTM-Cluster-Anwendungen. Zusätzliche Informationen zur Generierung von UTM-Cluster-Anwendungen finden Sie im Kapitel „Hinweise zur Generierung einer UTM-Cluster-Anwendung“ auf Seite 63.

Aus KDCROOT, Anwender-Teilprogrammen, Formaten und weiteren Anwendungsbestandteilen wie UTM-Systemmodule, Laufzeitsysteme der Programmiersprachen, Datenbank-Verbindungsmodule etc. müssen Sie das Anwendungsprogramm erzeugen.

Nähere Informationen dazu, wie Sie aus den ROOT-Tabellen und Anwendungskomponenten ein Anwendungsprogramm erzeugen, finden Sie im jeweiligen openUTM-Handbuch „Einsatz von openUTM-Anwendungen“.

Informationen zum Erstellen von Anwender-Teilprogrammen finden Sie in den openUTM-Handbüchern „Anwendungen programmieren mit KDCS“ und „Anwendungen erstellen mit X/Open-Schnittstellen“.

BB

Informationen dazu, wie Sie Formate im BS2000/OSD erstellen, finden Sie in den Handbüchern zu FHS.

2.1 UTM-Anwendung konfigurieren

Das Anwendungsprogramm benötigt zum Ablauf Informationen über:

- Anwendungseigenschaften
- UTM-Benutzerkennungen und Zugriffsschutz
- Eigenschaften von Clients und Druckern
- Eigenschaften von Partner-Anwendungen (Server-Anwendungen)
- Eigenschaften von Services, d.h. von Transaktionscodes und Teilprogrammen
- Service-gesteuerte Queues (Benutzer-, TAC- und temporäre Queues)
- Struktur der Anwendung (Aufteilung in Lademodule, Shared Objects bzw. DLLs)
- Tabellenplatzreservierungen in UTM-Objekttabellen für die dynamische Konfigurierung

Die Summe dieser Eigenschaften, die mit KDCDEF-Steueranweisungen definiert werden, heißt Konfiguration. Die KDCDEF-Steueranweisungen dienen als Input für das Generierungstool KDCDEF.
Eine Übersicht der KDCDEF-Steueranweisungen, jeweils zu Funktionsgruppen zusammengefasst, finden Sie auf Seite 244ff.

Die Konfigurationsinformationen und damit alle Verwaltungsdaten, die für den Anwendungsbetrieb notwendig sind, werden in der KDCFILE gespeichert.

Generieren mit WinAdmin

WinAdmin stellt Ihnen eine grafische Benutzeroberfläche zur Verfügung, mit der Sie Generierungsdaten (alle UTM-Objekte und Anwendungsparameter) komfortabel definieren, ändern und ergänzen können. Dabei können Sie auch mehrere Anwendungen gleichzeitig bearbeiten. Sie haben die Möglichkeit:

1. bestehende Generierungen von UTM-Anwendungen nach WinAdmin zu importieren und über die grafische Benutzeroberfläche zu bearbeiten;
   die Generierungsdaten werden in der Konfigurationsdatenbank abgespeichert.

   Da die Anwendungsdaten über die Administrations-Programmschnittstelle übergeben werden, können bestimmte Objekttypen nicht importiert werden (AREA, DATABASE, ENTRY, EXIT, MODULE, MPOOL, RMXA, SATSEL, TACCLASS, TCBENTRY, TLS, ULS); diese müssen Sie nachträglich über die WinAdmin-Oberfläche eintragen.

   Darüberhinaus können auch die Generierungsinformationen für die KDCDEF-Steueranweisungen FORMSYS und ROOT (Anwendungsparameter) nicht importiert werden.

   Fehlende Generierungsdaten können Sie über den Teilbaum „Generierung" ergänzen und verändern.

2. neue UTM-Anwendungen vollständig in WinAdmin zu generieren;
   die Generierungsdaten werden in der Konfigurationsdatenbank abgespeichert.

Um für OSI TP und LU6.1-Verbindungen zwischen zwei UTM-Anwendungen zu generieren, steht Ihnen ein geführter Dialog zur Verfügung. In diesem Dialog werden alle für die Verbindung relevanten Parameter aufeinander abgestimmt. Eigenschaften, die nur lokal von Bedeutung sind, müssen Sie ggf. über den Eigenschaftsdialog des jeweiligen Objekts nachtragen.

3. Aus den Generierungsdaten der Konfigurationsdatenbank können Sie eine Eingabedatei mit KDCDEF-Anweisungen erzeugen und als KDCDEF-Inputdateien auf den Anwendungsrechner zu transferieren.
4. KDCDEF- und KDCUPD-Läufe auf dem Anwendungsrechner zu starten;
   WinAdmin stellt das Protokoll des KDCDEF- bzw. KDCUPD-Laufs zur Verfügung.

Voraussetzung für die Möglichkeiten 1, 3 und 4 ist der Einsatz von *open*FT oder ftp auf den beteiligten Rechnern. Möglichkeit 4 funktioniert nur mit der Folgeverarbeitung von *open*FT. Wenn Sie *open*FT verwenden, muss sowohl auf dem WinAdmin- als auch auf dem Anwendungsrechner *open*FT ≥ V6.0 installiert sein.

Nähere Informationen zum Generieren von UTM-Anwendungen mit WinAdmin finden Sie im openUTM-Handbuch „Konzepte und Funktionen“ und in der Online-Hilfe zu WinAdmin.

2.2 Anwendungskomponenten generieren - Ergebnis des KDCDEF-Laufs

Die KDCFILE und die ROOT-Tabellen-Sourcen können wahlweise in einem KDCDEF-Lauf oder jeweils separat in unterschiedlichen KDCDEF-Läufen generiert werden. Mit der KDCDEF-Anweisung OPTION legen Sie fest, welche Generierungsobjekte KDCDEF erzeugen soll:

```
OPTION...,GEN={ KDCFILE | ROOTSRC | NO | ALL }
```

Den Namen der ROOT-Tabellen legen Sie mit der ROOT-Anweisung fest.

```
ROOT rootname
```

B Unter BS2000/OSD ist *rootname* der Name des ROOT-Tabellenmoduls.

X/W X/W Auf Unix- und Windows-Systemen ist *rootname* ein Namensteil der ROOT-Tabellen-Source (ROOTSRC).

KDCDEF liest die Steueranweisungen von der Standardeingabe oder aus einer Datei ein.

B B B Standardeingabe bedeutet unter BS2000/OSD SYSDTA (mit dem SDF-Kommando /ASSIGN-SYSDTA können Sie SYSDTA z.B. einer SAM- oder ISAM-Datei, einem Bibliothekselement vom Typ S einer PLAM-Bibliothek oder *SYSCMD zuweisen).

X/W X/W Auf Unix- und Windows-Systemen bedeutet Standardeingabe *stdin* (d.h. von der Unix- bzw. DOS-Kommandoebene).

Wie Sie KDCDEF starten und die Steueranweisungen an KDCDEF übergeben, ist auf Seite 251 detailliert beschrieben.

KDCDEF führt für alle KDCDEF-Anweisungen Syntax- und Plausibilitätsprüfungen durch. Erkennt KDCDEF dabei keine schwerwiegenden Fehler, dann werden für eine stand-alone UTM-Anwendung die in Bild 2 auf Seite 34 aufgeführten Dateien erstellt.

Welche Dateien erstellt werden, wenn eine UTM-Cluster-Anwendung generiert wird, entnehmen Sie Bild 5 auf Seite 65.

Auch wenn mit einem KDCDEF-Lauf – gesteuert durch die OPTION-Anweisung – nur ein Teil der Konfiguration (neu) erstellt werden soll, geben Sie dennoch für jeden Generierungslauf die Anweisungen für die gesamte Konfiguration an. Nur dann ist KDCDEF in der Lage, die Vollständigkeit und Konsistenz der Generierungsanweisungen zu überprüfen.

KDCDEF führt Plausibilitätsprüfungen immer für alle Anweisungen durch. Soll in einem KDCDEF-Lauf z.B. nur eine ROOT-Source erzeugt werden, dann prüft KDCDEF auch die Anweisungen, die sich nur auf die KDCFILE auswirken.

Durch diese vollständige Prüfung können Inkonsistenzen beim Erzeugen von ROOT-Tabellenmodul und KDCFILE, die sonst erst beim Start der Anwendung entdeckt würden, frühzeitig erkannt und damit Folgefehler vermieden werden.

Bild 2 zeigt, welche Dateien erstellt werden, wenn Sie eine stand-alone UTM-Anwendung definieren.

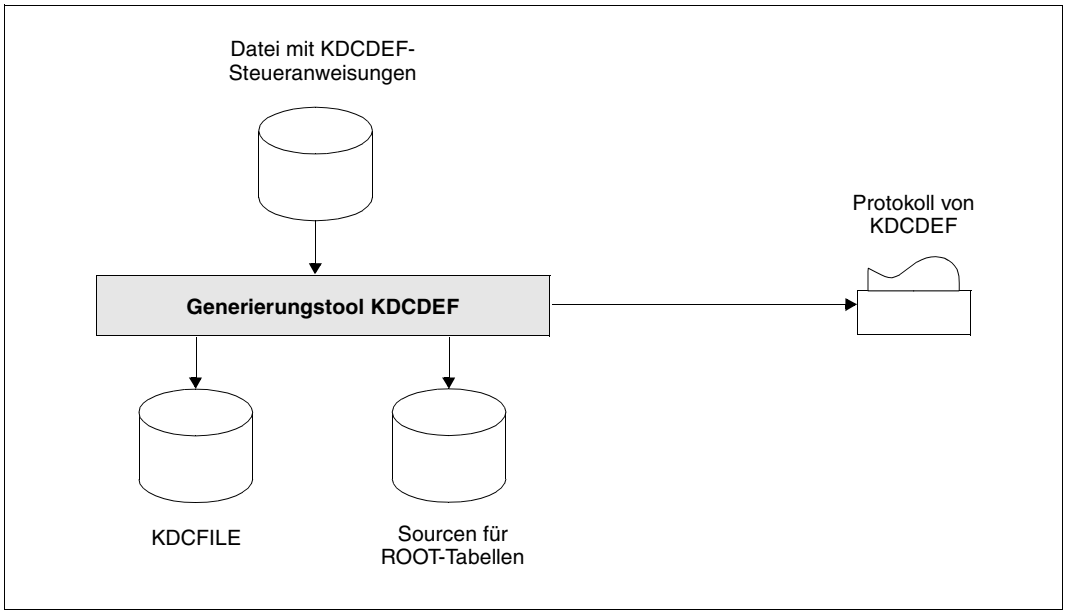


Bild 2: Ergebnis des KDCDEF-Laufs (bei OPTION ...,GEN=ALL) für eine stand-alone UTM-Anwendung

KDCDEF-Anweisungen für eine Minimal-Konfiguration

Damit Ihre UTM-Anwendung ablaufen kann, müssen Sie mindestens die folgenden Steueranweisungen an KDCDEF übergeben.

Für die verteilte Verarbeitung, den Anschluss von Druckern etc. müssen Sie weitere KDCDEF-Anweisungen absetzen. Nähere Informationen dazu finden Sie in Abschnitt „Verteilte Verarbeitung über das LU6.1-Protokoll“ auf Seite 72, Abschnitt „Verteilte Verarbeitung über das OSI TP-Protokoll“ auf Seite 88 und Abschnitt „Drucker generieren (auf BS2000/OSD und Unix-Systemen)“ auf Seite 165.

Bei den mit '*' gekennzeichneten Zeilen handelt es sich um Kommentare.

Minimalkonfiguration für BS2000/OSD

```
*******************************************************************************
* Angabe, welche Teile des Anwendungsprogramms KDCDEF erzeugen soll           *
*******************************************************************************

OPTION GEN=...

*******************************************************************************
* Name der Root-Tabellen festlegen                                            *
*******************************************************************************

ROOT applroot

*******************************************************************************
* Anwendungsparameter festlegen                                               *
*******************************************************************************

* Anwendungsname, mit dem die Anwendung gestartet wird, bzw. von Clients
* angesprochen werden kann

MAX APPLINAME=sample

* Basisname der Anwendung festlegen.

MAX KDCFILE=filebase

* Maximalanzahl der Prozesse der UTM-Anwendung festlegen:

MAX TASKS=2
```

```
B  ********************************************************************************
B  * optional: Datenbank-System generieren (im Beispiel Oracle)
B  *
B  ********************************************************************************

B  * BS2000/OSD:

B  DATABASE TYPE=XA
B

B  ********************************************************************************
B  * optional: Verwendetes Formatierungssystem angeben                            *
B  ********************************************************************************

B  * die Anweisung FORMSYS müssen Sie nur angeben, falls Ihre UTM-Anwendung
B  * im formatierten Betrieb ablaufen soll

B  FORMSYS TYPE=FHS
B

B  ********************************************************************************
B  * optional: Anschlusspunkte (LTERM-Partner) für Clients/TS-Anwendungen      *
B  ********************************************************************************

B  * Z.B. offene LTERM-Pools generieren, damit sich Clients/TS-Anwendungen
B  * an die Anwendung anschließen können

B  * LTERM-Pools für die verschiedenen Client-Typen ------------------------ (1)

B  TPOOL LTERM=client,NUMBER=...,PTYPE=*ANY,PRONAM=*ANY
B  TPOOL LTERM=clientr,NUMBER=...,PTYPE=UPIC-R,PRONAM=*ANY
B  TPOOL LTERM=appli,NUMBER=...,PTYPE=APPLI,PRONAM=*ANY
B  TPOOL LTERM=socket,NUMBER=...,PTYPE=SOCKET,PRONAM=*ANY

B  ********************************************************************************
B  * Services generieren                                                          *
B  ********************************************************************************

B  * Eigene Teilprogramme, die die Services erbringen, und die zugehörigen
B  * Transaktionscodes generieren
B  * (für COMP=... ist der verwendete Compiler bzw. das Laufzeitsystem, meistens
B  * „ILCS“, anzugeben)

B  PROGRAM=userpu,COMP= ...
B  TAC usertc,PROGRAM=userpu, ... ------------------------------------------- (2)
B
```

```
B  ********************************************************************************
B  * optional: Administration                                                      *
B  ********************************************************************************

B  * Administrations-Programm KDCADM

B  PROGRAM KDCADM,COMP=ILCS

B  * Administrationskommando KDCSHUT generieren, damit die Anwendung immer
B  * normal beendet werden kann

B  TAC KDCSHUT,PROGRAM=KDCADM ... ------------------------------------------ (3)

B  * In Anwendungen mit Benutzerkennungen: Benutzerkennung für den Administrator

B  USER admin,PERMIT=ADMIN,PASS=....

B  * Soll die Administration über openUTM-WinAdmin erfolgen,
B  * müssen Sie die folgende TAC- und PROGRAM Anweisung absetzen
B  * und einen Anschluss für den UPIC-Client (hier einen LTERM-Pool) generieren
B  * Außerdem sollten Sie dann die administrationsberechtigte Benutzerkennung
B  * admin ohne Wiederanlauf-Eigenschaft oder eine eigene
B  * admistrationsberechtigte Benutzerkennung ohne Wiederanlauf-Eigenschaft für
B  * die Adminstration über openUTM-WinAdmin generieren.

B  * Administrations-Programm KDCWADMI

B  PROGRAM KDCWADMI,COMP=ILCS
B  TAC KDCWADMI,PROGRAM=KDCWADMI,CALL=BOTH,ADMIN=Y

B  TPOOL LTERM=WADM,PTYPE=UPIC-R,PRONAM=*ANY, NUMBER=1
B

B  ********************************************************************************
B  * optional: Tabellenplätze für die dynamische Administration reservieren      *
B  ********************************************************************************

B  RESERVE OBJECT=...,NUMBER=... ------------------------------------------- (4)
B
B  END
B
```

Anmerkungen

(1) Für jeden Client-Typ, der sich an die Anwendung anschließen kann (Terminal, UPIC-Client, TS-Anwendung), müssen Sie einen eigenen LTERM-Pool generieren. Für Terminals reicht ein LTERM-Pool - unabhängig vom Typ der Terminals, die sich an die Anwendung anmelden sollen. Sie können die LTERM-Pools auch so generieren, dass sich alle Clients eines bestimmten Typs anmelden können - unabhängig von dem Rechner, an dem sie sich befinden.

Den Anschluss von Clients können Sie auch mit Hilfe von LTERM-/PTERM-Anweisungen realisieren. LTERM-/PTERM-Anweisungen müssen Sie insbesondere dann verwenden, wenn die UTM-Anwendung selbst Verbindungen zu Clients (z.B. TS-Anwendungen) aufbauen oder wenn ein Drucker generiert werden soll.

(2) Sie können einem Teilprogramm auch mehrere Transaktionscodes zuordnen, wenn es mehrere verschiedene Services erbringt.

(3) Durch weitere TAC-Anweisungen können Sie alle Administrationskommandos generieren, die Sie im Betrieb verwenden wollen. Wollen Sie für die Administration der Anwendung eigene Administrationsprogramme verwenden, dann müssen Sie diese mit entsprechenden PROGRAM- und TAC-Anweisungen generieren.

(4) Mit Hilfe der Administration können Sie im laufenden Betrieb der Anwendung weitere Objekte in die Konfiguration der Anwendung eintragen (siehe openUTM-Handbuch „Anwendungen administrieren“). Für diese Objekte müssen Sie bei der KDCDEF-Generierung Tabellenplätze in der KDCFILE erzeugen.

X/W *Minimalkonfiguration für Unix- und Windows-Systeme*

```
X/W ********************************************************************************
X/W * Angabe, welche Teile des Anwendungsprogramms KDCDEF erzeugen soll           *
X/W ********************************************************************************

X/W OPTION GEN=...
X/W

X/W ********************************************************************************
X/W * Name der Root-Tabellen festlegen                                            *
X/W ********************************************************************************

X/W ROOT applroot
X/W

X/W ********************************************************************************
X/W * Anwendungsparameter festlegen                                               *
X/W ********************************************************************************

X/W * Anwendungsname, mit dem die Anwendung gestartet wird, bzw. von Clients
X/W * angesprochen werden kann

X/W MAX APPLINAME=sample

X/W * Basisverzeichnis der Anwendung festlegen.
X/W * Das ist das Verzeichnis, in dem u.a. die KDCFILE abgelegt wird.

X/W MAX KDCFILE=filebase

X/W * Schlüssel für Shared Memory Bereiche festlegen

X/W MAX CACHESHMKEY=...,IPCSHMKEY=...,KAASHMKEY=...
X/W                [,OSISHMKEY=...,XAPTPSHMKEY=...] ---------------------- (1)
X/W
X/W Semaphore Schlüssel für die anwendungsglobalen Semaphore festlegen
X/W MAX SEMARRAY=number,number1
X/W
X/W * Maximalanzahl der Prozesse der UTM-Anwendung festlegen.
X/W
X/W MAX TASKS=2
X/W

X/W ********************************************************************************
X/W * optional: Datenbank-System generieren (am Beispiel Oracle)                  *
X/W ********************************************************************************

X/W RMXA XASWITCH=xaoswd,SPEC=C
X/W
```

```
X/W  ********************************************************************************
X/W  * optional: Anschlusspunkte (LTERM-Partner) für Clients/TS-Anwendungen      *
X/W  ********************************************************************************

X/W  * Z.B. offene LTERM-Pools generieren, damit sich Clients/TS-Anwendungen an  *
X/W  * die Anwendung anschließen können

X/W  * LTERM-Pools für die verschiedenen Client-Typen ------------------------ (2)

X/W  TPOOL LTERM=CLIENTR,PTYPE=UPIC-R,NUMBER=...
X/W  TPOOL LTERM=CLIENTL,PTYPE=UPIC-L,NUMBER=...
X/W  TPOOL LTERM=APPLI,PTYPE=APPLI,NUMBER=...
X/W  TPOOL LTERM=SOCKET,PTYPE=SOCKET,NUMBER=...
X/W  TPOOL LTERM=TERM,PTYPE=TTY,NUMBER=...
X/W

X/W  ********************************************************************************
X/W  * Services generieren                                                         *
X/W  ********************************************************************************

X/W  * Eigene Teilprogramme, die die Services erbringen, und die zugehörigen
X/W  * Transaktionscodes generieren
X/W  * (für COMP=... ist der verwendete Compiler anzugeben)

X/W  PROGRAM=userpu,COMP=...
X/W  TAC usertc,PROGRAM=userpu,... ------------------------------------------- (3)
X/W
```

```
X/W ********************************************************************************
X/W * optional: Administration                                                     *
X/W ********************************************************************************

X/W * Administrations-Programm KDCADM

X/W PROGRAM KDCADM,COMP=C

X/W * Administrationskommando KDCSHUT generieren, damit die Anwendung immer
X/W * normal beendet werden kann

X/W TAC KDCSHUT,PROGRAM=KDCADM ... ------------------------------------------ (4)

X/W * In Anwendungen mit Benutzerkennungen: Benutzerkennung für den Administrator

X/W USER admin,PERMIT=ADMIN,PASS=....

X/W * Soll die Administration über openUTM-WinAdmin erfolgen,
X/W * müssen Sie die folgende TAC- und PROGRAM Anweisung absetzen
X/W * und einen Anschluss für den UPIC-Client (hier einen LTERM-Pool) generieren

X/W * Administrations-Programm KDCWADMI

X/W PROGRAM KDCWADMI,COMP=C
X/W TAC KDCWADMI,PROGRAM=KDCWADMI,CALL=BOTH,ADMIN=Y

X/W TPOOL LTERM=WADM,PTYPE=UPIC-R,NUMBER=1
X/W

X/W ********************************************************************************
X/W * optional: Tabellenplätze für die dynamische Administration reservieren    *
X/W ********************************************************************************

X/W RESERVE OBJECT=...,NUMBER=... -------------------------------------------- (5)
X/W
X/W END
X/W
```

X/W *Anmerkungen*

X/W (1) Die Shared-Memory-Schlüssel OSISHMKEY= und XAPTPSHMKEY= müssen Sie nur angeben, wenn Sie Objekte für die Kommunikation über OSI TP generieren. Die anderen Shared Memories benötigt jede UTM-Anwendung, die unter Unix- oder Windows-Systemen abläuft.

X/W X/W X/W X/W X/W (2) Unter Unix- oder Windows-Systemen müssen Sie für jeden Client-Typ, der sich an die Anwendung anschließen kann (Terminal, UPIC-Client, TS-Anwendung), einen eigenen LTERM-Pool generieren. Sie können die LTERM-Pools auch so generieren, dass sich alle Clients eines bestimmten Typs anmelden können - unabhängig von dem Rechner, an dem sie sich befinden.

X/W X/W X/W X/W X/W Den Anschluss von Clients können Sie auch mit Hilfe von LTERM-/PTERM-Anweisungen realisieren. LTERM-/PTERM-Anweisungen müssen Sie insbesondere dann verwenden, wenn die UTM-Anwendung selbst Verbindungen zu Clients (z.B. TS-Anwendungen) aufbauen soll oder wenn auf Unix-Systemen ein Drucker generiert werden soll.

X/W X/W (3) Sie können einem Teilprogramm auch mehrere Transaktionscodes zuordnen, wenn es mehrere verschiedene Services erbringt.

X/W X/W X/W X/W (4) Durch weitere TAC-Anweisungen können Sie alle Administrationskommandos generieren, die Sie im Betrieb verwenden wollen. Wollen Sie für die Administration der Anwendung eigene Administrationsprogramme verwenden, dann müssen Sie diese mit entsprechenden PROGRAM- und TAC-Anweisungen generieren.

X/W X/W X/W X/W (5) Mit Hilfe der Administration können Sie im laufenden Betrieb der Anwendung weitere Objekte in die Konfiguration der Anwendung eintragen (siehe openUTM-Handbuch „Anwendungen administrieren“). Für diese Objekte müssen Sie bei der KDCDEF-Generierung Tabellenplätze in der KDCFILE erzeugen.

Bestehende UTM-Anwendung neu generieren

Wenn Sie für eine bestehende Anwendung (d.h. KDCROOT und KDCFILE existieren bereits) eine neue ROOT-Tabellen-Source und/oder eine neue KDCFILE generieren wollen, müssen Sie Folgendes beachten:

Informationen zu Objekten, die im Betrieb der Anwendung dynamisch in die KDCFILE eingetragen oder deren Eigenschaften geändert wurden, müssen Sie in die neue KDCFILE übernehmen. Dazu steht Ihnen die Funktion „inverser KDCDEF“ zur Verfügung. Mit ihr erzeugen Sie aus den Konfigurationsinformationen der aktuellen KDCFILE Steueranweisungen für den KDCDEF-Lauf, die Sie sofort weiterverarbeiten können. Im KDCDEF-Lauf müssen Sie dazu die Steueranweisung CREATE-CONTROL-STATEMENTS aufrufen.

Über die UTM-Administration können Sie den inversen KDCDEF-Lauf auch im laufenden Betrieb der Anwendung durchführen.

Informationen zur Funktion „inverser KDCDEF“ finden Sie im Abschnitt „Inverser KDCDEF“ auf Seite 265.

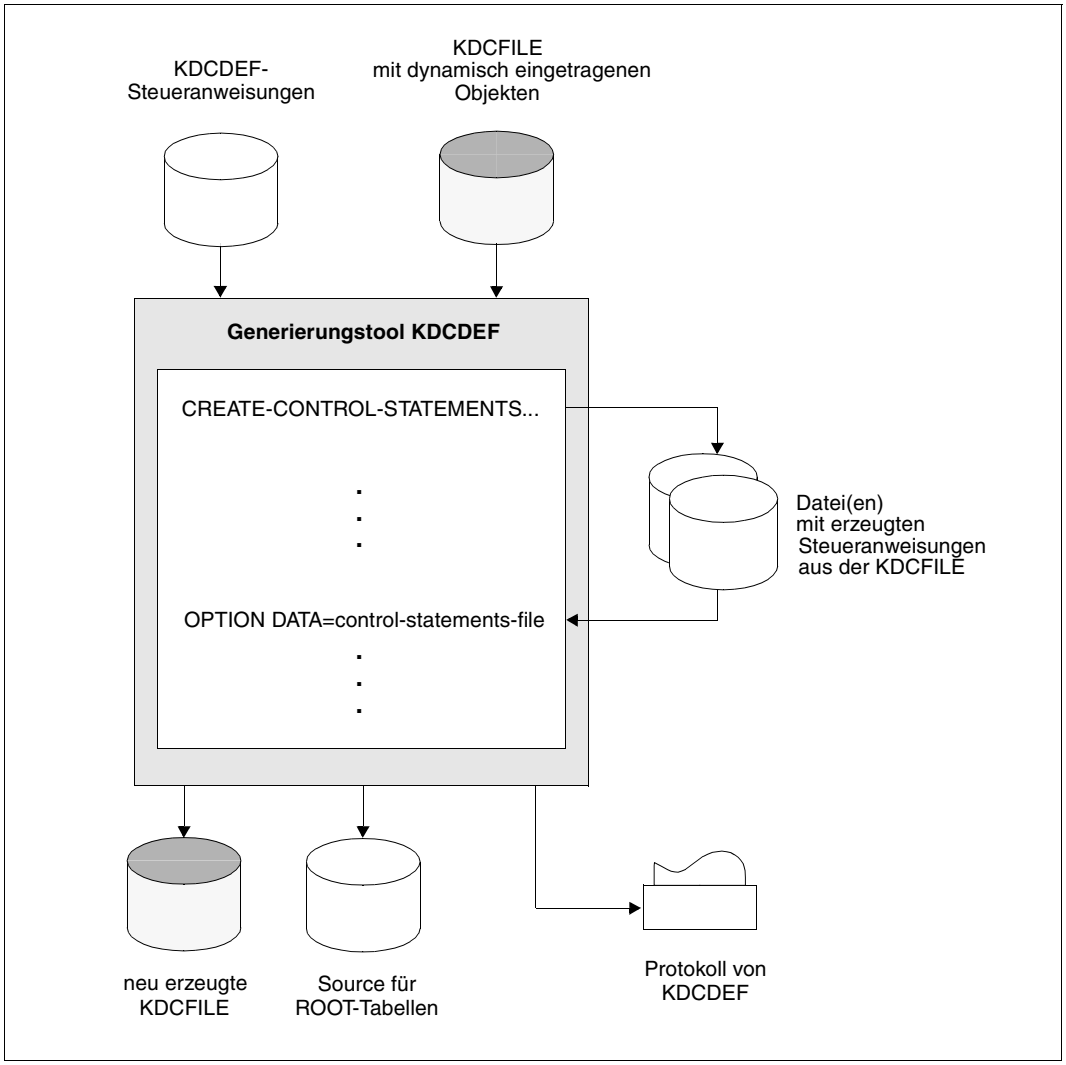


Bild 3: KDCDEF-Lauf mit inversem KDCDEF

2.3 Die KDCFILE

Die KDCFILE ist eine Datei, die alle für den Ablauf einer UTM-Anwendung notwendigen Daten enthält. Beim Betrieb der Anwendung wird sie von allen Prozessen der Anwendung gemeinsam genutzt. Im einfachsten Fall besteht die KDCFILE aus genau einer Datei (unter BS2000/OSD ist dies eine PAM-Datei). Die KDCFILE kann auch auf mehrere Dateien verteilt und/oder aus Sicherheitsgründen doppelt geführt werden.

Die KDCFILE ist logisch in folgende drei Bereiche gegliedert:

- **Verwaltungsdaten** siehe Seite 48
- **Pagepool** siehe Seite 48
- **Wiederanlaufbereich** siehe Seite 51

KDCDEF-Generierung

Die KDCFILE wird im KDCDEF-Lauf erzeugt, wenn Sie folgende KDCDEF-Anweisung angeben:

```
OPTION...,GEN=KDCFILE oder GEN=ALL
```

Folgende Eigenschaften der KDCFILE müssen Sie bei der KDCDEF-Generierung festlegen:

- Blockung der Daten
  Jeder Bereich der KDCFILE ist in Einheiten von wahlweise je 2 K bzw. 4 K organisiert. Man bezeichnet diese Einheiten als UTM-Seiten der KDCFILE. Die Blockung einer UTM-Seite legen Sie fest mit der KDCDEF-Steueranweisung:

  ```
  MAX...,BLKSIZE={ 2K | 4K }
  ```

  Ob eine Blockung von 2K oder 4K günstiger ist, ist abhängig von der Größe der Datenbereiche (GSSB, LSSB usw.) und den Nachrichtenlängen, die in Ihrer Anwendung auftreten. Siehe dazu auch den Abschnitt „Pagepool“ auf Seite 48.

- Basisname der KDCFILE
  Den Basisnamen (im Folgenden *filebase* genannt) und die einfache oder doppelte Dateiführung der KDCFILE legen Sie fest mit:

  ```
  MAX..., KDCFILE={ filebase [, SINGLE | DOUBLE ] }
  ```

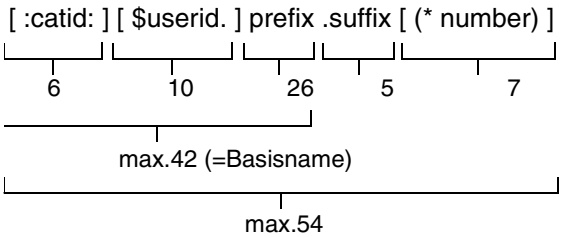
  Bei doppelter Dateiführung ist der Inhalt beider KDCFILE-Dateien immer identisch. Bei Zerstörung einer der beiden Dateien kann die zerstörte Datei durch einfaches Kopieren der nicht zerstörten Datei wiederhergestellt werden.

Der in *filebase* angegebene Name ist auch der Namensbestandteil weiterer Dateien und Dateigenerationen der Anwendung (z.B. System- und Benutzer-Protokolldatei). *filebase* ist damit der so genannte Basisname der Anwendung.

Der Basisname *filebase* hat auf den einzelnen Plattformen folgende Eigenschaften:

B *BS2000/OSD:*

B Die vollständigen Dateinamen der aus *filebase* abgeleiteten Dateien setzen sich zusammen aus:

B 

catid	Katalogkennung der KDCFILE
userid	BS2000-Benutzerkennung
prefix	frei wählbarer Namensteil
suffix	Namensteil, der von openUTM vergeben wird und die Funktion der jeweiligen Datei kennzeichnet
(*number)	kennzeichnet ggf. eine Dateigenerationsgruppe

B Die Gesamtlänge des Dateinamens darf 54 Zeichen nicht überschreiten. Der Basisname *filebase* darf einschließlich *userid* und *catid* max. 42 Zeichen lang sein. Wird bei der Definition des Basisnamens auf die Angabe von *catid* und *userid* verzichtet (*filebase*=*prefix*), so müssen deren Längen trotzdem für die Berechnung der Gesamtlänge des Dateinamens berücksichtigt werden.

B Wird in der Anwendung ohne Dateigenerationsgruppen gearbeitet (d.h. ohne USLOG-Dateien und SYSLOG-Dateigenerationen), darf *prefix* maximal 33 Zeichen lang sein. Im anderen Fall darf *prefix* 26 Zeichen nicht überschreiten.

B Das Generierungstool KDCDEF erzeugt im KDCDEF-Lauf die folgenden Dateien:

B – *filebase*.KDCA bei einfacher Dateiführung (SINGLE)
B – *filebase*.KDCA und *filebase*.KDCB bei doppelter Dateiführung (DOUBLE).

B Bei Aufspaltung der KDCFILE werden weitere Dateien erzeugt, siehe
B Abschnitt „KDCFILE aufspalten“ auf Seite 56.

X/W *Unix- und Windows-Systeme:*

X/W *filebase* ist der Name des Basisverzeichnisses, in dem die KDCFILE abgelegt wird.

X/W KDCDEF richtet die KDCFILE unter dem Dateiverzeichnis *filebase* ein. *filebase* ist dabei
X/W der vollqualifizierte Name des Dateiverzeichnisses, das **vor** dem KDCDEF-Lauf einge-
X/W richtet werden muss!

X/W Das Generierungstool KDCDEF erzeugt im KDCDEF-Lauf die folgenden Dateien im
X/W Dateiverzeichnis *filebase*:

X/W – KDCA bei einfacher Dateiführung (SINGLE)
X/W – KDCA und KDCB bei doppelter Dateiführung (DOUBLE).

X/W Das Dateiverzeichnis *filebase* enthält neben der KDCFILE in der Regel noch weitere Da-
X/W teien und Dateiverzeichnisse:

X/W – Dateien des Pagepools bzw. Wiederanlaufbereichs bei aufgespalteter KDCFILE
X/W – Dateiverzeichnisse für die Benutzerprotokoll-Dateien (USLOG-Dateien)
X/W – System-Protokolldatei (SYSLOG-Datei) bzw. das Verzeichnis SYSLOG mit Gene-
X/W rationen von Systemprotokolldateien
X/W – ROOT-Tabellen-Source
X/W – Anwendungsprogramm
X/W – Dateiverzeichnis DUMP: In dieses Verzeichnis werden die Dump-Dateien geschrie-
X/W ben sowie temporäre KTA-TRCA-Dateien, die beim PEND-ERROR und
X/W Programmaustausch kurzzeitig entstehen.

- Größe des Pagepools
  Die Größe des Pagepools legen Sie fest mit:

  ```
  MAX...,PGPOOL=( number, warnlevel1, warnlevel2 )
  ```

  Sehen Sie dazu auch den Abschnitt „Pagepool“ auf Seite 48.

- Größe des Wiederanlaufbereichs
  Die Größe des Puffers und des Wiederanlaufbereichs legen Sie fest mit:

  ```
  MAX ...,RECBUF=( number,length )
  ```

  Sehen Sie dazu auch den Abschnitt „Wiederanlaufbereich“ auf Seite 51.

Pagepool und den Wiederanlaufbereich kann man bei der Generierung auch auf mehrere Dateien aufteilen. Sehen Sie dazu den Abschnitt „KDCFILE aufspalten“ auf Seite 56.

Datensicherheit - Doppelte KDCFILE-Führung

Unter dem Gesichtspunkt der erhöhten Sicherheit kann es sinnvoll sein, die KDCFILE doppelt zu führen. Wird eine der Dateien zerstört, dann können Sie ohne Datenverlust mit der anderen KDCFILE weiterarbeiten.

Die Zeiten für die Ein-/Ausgaben erhöhen sich bei doppelter KDCFILE-Führung nicht wesentlich (sie verdoppeln sich keinesfalls!) und bewirken deshalb auch keine besondere Performancebelastung.

Bei doppelter KDCFILE-Führung ist es sinnvoll, beide Dateien auf unterschiedlichen Datenträgern (Platten) abzulegen, damit bei physikalischer Zerstörung eines Datenträgers eine weiterverwendbare Kopie erhalten bleibt.

B *BS2000/OSD:*

B Sie können die beiden Dateien durch entsprechende /CREATE-FILE-Kommandos vor dem
B KDCDEF-Lauf oder durch ein anschließendes Kopieren auf die gewünschten Datenträger
B verteilen. Beim Generieren der Anwendung können Sie außerdem mit dem Parameter
B CATID der MAX-Anweisung den beiden Dateien unterschiedliche CATIDs zuordnen.

B Die KDCFILE wird doppelt geführt, wenn Sie bei der KDCDEF-Generierung
B `MAX KDCFILE=(....,DOUBLE)` angeben.

X *Unix-Systeme:*

X Auf Unix-Systemen können Sie die beiden KDCFILEs auf verschiedene Platten legen. Das
X ist i.A. nur mit Hilfe von symbolischen Verweisen (ln -s) auf raw-devices oder über Datei-
X systemgrenzen hinweg möglich. So bleibt bei physikalischer Zerstörung eines Datenträ-
X gers eine weiterverwendbare Kopie erhalten.

X Die KDCFILE wird doppelt geführt, wenn Sie bei der KDCDEF-Generierung
X `MAX KDCFILE=(....,DOUBLE)` angeben.

W *Windows-Systeme:*

W Auf Windows-Systemen können Sie außerdem eine erhöhte Datensicherheit mit Betriebs-
W system-Mitteln erreichen. Sie können z.B. die KDCFILE einfach führen
W (`MAX KDCFILE=(....,SINGLE)`) und von dem Datenträger, auf dem sich die KDCFILE befin-
W det, auf einer anderen Platte einen sogenannten Spiegelsatz erstellen. Während des Be-
W triebs werden alle Änderungen an der KDCFILE auch auf dem Spiegelsatz durchgeführt.
W Bei physikalischer Zerstörung eines Datenträgers kann ohne Datenverlust mit der anderen
W Platte weitergearbeitet werden.

2.3.1 Verwaltungsdaten

Der Bereich der Verwaltungsdaten enthält die Konfigurationsinformationen wie Betriebsparameter der Anwendung, Inhaltsverzeichnisse für alle über Namen ansprechbaren Objekte, Verwaltungsinformation für Pagepool und Wiederanlaufbereich sowie Tabellen für Benutzerkennungen, Clients, LTERM-Partner, Transaktionscodes, Key- und Lockcodes und Funktionstasten.

Mit den Verwaltungsdaten arbeiten alle Tasks/Workprozesse der Anwendung und tauschen über sie untereinander Informationen aus.

Die Verwaltungsdaten werden durch das Generierungstool KDCDEF initialisiert. Sie werden beim Start der Anwendung in einen gemeinsamen Speicher gelesen, auf den alle Tasks/Workprozesse der Anwendung zugreifen können.

B Unter BS2000/OSD liegt dieser Speicherbereich in einem Common Memory Pool.

X/W X/W Auf Unix- oder Windows-Systemen werden die Verwaltungsdaten in ein Shared Memory Segment gebracht.

In einer UTM-S-Anwendung schreibt openUTM die Verwaltungsdaten mit den Änderungen, die inzwischen erfolgt sind, in bestimmten Abständen auf die KDCFILE zurück (Periodic Write). Das gleiche geschieht am Ende des Anwendungslaufes. Dieser Stand der Verwaltungsdaten ist die Basis für den nächsten Lauf der Anwendung.

In einer UTM-F-Anwendung schreibt openUTM nur bestimmte geänderte Verwaltungsdaten auf die KDCFILE zurück, wie beispielsweise geänderte Benutzerpasswörter und Generierungsdaten, die per dynamischer Administration eingebracht wurden.

2.3.2 Pagepool

Im Pagepool werden Benutzerdaten gespeichert, die während des Anwendungslaufs entstehen. Das sind:

- LSSBs, GSSBs, TLS- und ULS-Blöcke
- Message Queues, d.h. Asynchron-Nachrichten (auch zeitgesteuerte) für Clients, Asynchron-Vorgänge und Service-gesteuerte Queues, unter anderem auch die Dead Letter Queue
- zwischengespeicherte Sätze der Benutzer-Protokolldatei (USLOG)
- Vorgangsdaten (KB-Programmbereich, letzte Dialog-Nachricht etc.)
- Dialog-Nachrichten, die als Folge der TAC-Klassen- bzw. Prioritätensteuerung nach Eingabe zwischengespeichert werden.
- Ausgabe-Nachrichten an Clients

Die laufende UTM-Anwendung greift auf den Pagepool über den UTM-Cache zu. In der KDCDEF-Generierung legen Sie die Größe des Pagepools (Anzahl der UTM-Seiten) in der MAX-Anweisung fest:

```
MAX...,PGPOOL=( number, warnlevel1, warnlevel2 )
```

number Größe des Pagepools in Anzahl UTM-Seiten

warnlevel1 Die erste Warnung wird ausgegeben, wenn die hier angegebene Belegung des Pagepools in Prozent erreicht ist.

warnlevel2 Die zweite Warnung wird ausgegeben, wenn die hier angegebene Belegung des Pagepools in Prozent erreicht ist.

Eine Möglichkeit, sich genauer über die Herkunft der im Pagepool gespeicherten Daten zu informieren, bietet das Tool KDCUPD. Dabei kann für jedes Objekt der Anwendung, z.B. für jeden Benutzer, die Anzahl der belegten Seiten angezeigt werden. Weitere Informationen dazu finden Sie im Kapitel „Das Tool KDCUPD - KDCFILE aktualisieren“ auf Seite 585.

Abschätzen der benötigten Größe des Pagepools

Die einmal festgelegte Größe des Pagepools kann bei laufender Anwendung nicht verändert werden. Es ist daher zu empfehlen, bereits beim Entwurf einer UTM-Anwendung die benötigte Pagepool-Größe im Betrieb abzuschätzen. Eine Änderung ist nur nach Beendigung der Anwendung möglich. Dazu generieren Sie mit dem Generierungstool KDCDEF die KDCFILE neu, wobei anschließend bestehende Benutzerdaten mit dem Tool KDCUPD aus der alten in die neue KDCFILE übernommen werden können. Weitere Informationen dazu finden Sie im Kapitel „Das Tool KDCUPD - KDCFILE aktualisieren“ auf Seite 585.

Um die Größe des Pagepools abschätzen zu können, muss man das Verhalten der Teilprogramme untersuchen und feststellen, welche Datenbereiche im Pagepool abgelegt werden und wie groß diese sind. Dabei ist zu beachten:

- Der Pagepool ist in UTM-Seiten unterteilt:
  Eine UTM-Seite ist entweder 2KB oder 4KB groß, abhängig vom Wert des Parameters BLKSIZE= in der MAX-Anweisung.
- Für die Datenbereiche GSSB, LSSB, TLS und ULS gilt:
  Jeder einzelne dieser Datenbereiche beginnt auf einer neuen UTM-Seite. Von jeder UTM-Seite stehen dem Anwender 1994 Byte (bei 2KB-UTM-Seiten) bzw. 4042 Byte (bei 4KB-UTM-Seiten) für Benutzerdaten zur Verfügung. Den Rest belegt openUTM.

- Für Asynchron-Nachrichten gilt:
  Jede Nachricht beginnt auf einer neuen UTM-Seite. Auf der ersten UTM-Seite einer Nachricht sind mindestens 1914 Byte (bei 2KB-UTM-Seiten) bzw. 3962 Byte (bei 4KB-UTM-Seiten) für Anwenderdaten nutzbar. Auf jeder Folgeseite, die die Nachricht belegt, sind mindestens 2030 Byte (bei 2KB-UTM-Seiten) bzw. 4078 Byte (bei 4KB-UTM-Seiten) für Anwenderdaten nutzbar.

  Es ist möglich, dass in Folgeversionen von openUTM pro UTM-Seite weniger Platz für Anwenderdaten zur Verfügung steht. Sie sollten deshalb bei der Programmierung die angegebenen Maximalwerte nicht voll ausschöpfen.
- Wird ein bereits existierender Bereich geändert, dann legt openUTM den neuen Bereich bis zum Ende der Transaktion an einer anderen Stelle im Pagepool ab. Somit existiert der Bereich kurzzeitig zweimal.

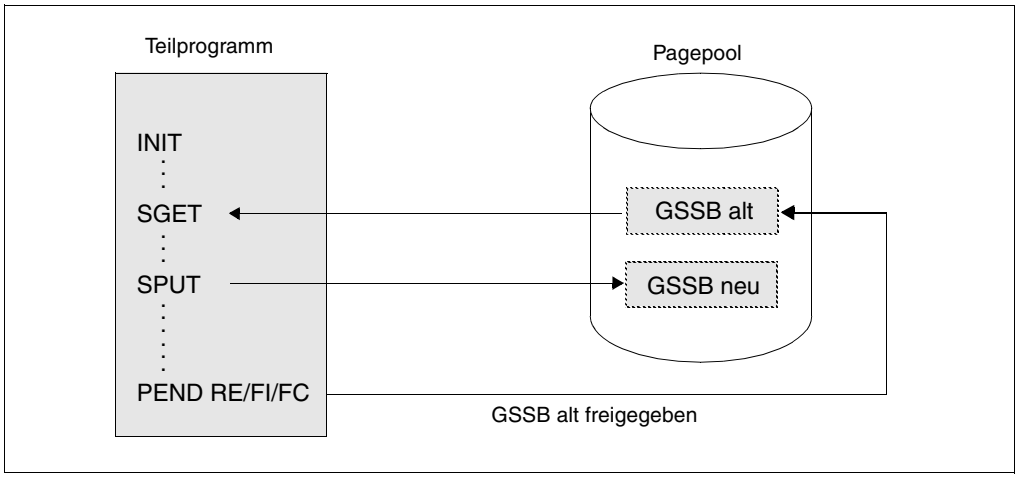


Bild 4: Doppelte Führung von geänderten Bereichen im Pagepool

Berücksichtigen Sie auch das Aufkommen an FPUT- und LPUT-Nachrichten. Wählen Sie daher den Pagepool nicht zu klein.

Warnungen vor einem Pagepool-Überlauf

Bei laufender Anwendung muss verhindert werden, dass der Pagepool ganz voll wird. Zu diesem Zweck ergreift openUTM folgende Schutzmaßnahmen:

- Es gibt zwei Warnstufen (Prozentwerte) für die Belegung des Pagepools, die per Generierung einstellbar sind. Werden diese Stufen über- bzw. unterschritten, erzeugt openUTM die Meldung K040 bzw. K041, auf die eine MSGTAC-Routine des Anwenders reagieren kann.
- Lokale Asynchron-Nachrichten sowie LPUT-Aufrufe zum Schreiben von Sätzen in die USLOG-Datei werden zurückgewiesen, wenn die Belegung des Pagepools Warnstufe 2 erreicht hat.
- Asynchron-Nachrichten von einer Partner-Anwendung über LU6.1 bzw. OSI TP werden abgelehnt, wenn die Belegung des Pagepools Warnstufe 2 erreicht hat. Die Verbindung wird abgebaut. Bei Kommunikation über OSI TP wird in der Auftraggeber- und Auftragnehmer-Anwendung jeweils die Meldung `K119 OSI-TP error information` mit dem Insert `DIA3=21` ausgegeben. Abhängig vom Wert in MAX CONRTIME wird zyklisch versucht, die Asynchron-Nachricht erneut an die Auftragnehmer-Anwendung zu schicken.
- Ein von einem Terminal oder einer TS-Anwendung erteilter Asynchron-Auftrag wird mit der Meldung K101 abgelehnt, wenn die Belegung des Pagepools Warnstufe 2 erreicht hat.

2.3.3 Wiederanlaufbereich

KDCS-Aufrufe in einem Teilprogramm haben Änderungen in den Verwaltungsdaten zur Folge. openUTM sammelt Informationen über alle Änderungen, die innerhalb einer Transaktion anfallen - also vom ersten INIT-Aufruf bis zum Transaktionsende - in einem Prozess-spezifischen Speicherbereich. Unter BS2000/OSD ist dies ein Puffer im Klasse-5-Speicher.

Bei Transaktionsende bildet openUTM bei einer UTM-S-Anwendung aus der Information in diesem Puffer einen Datensatz mit Wiederanlauf-Information und schreibt ihn in den Wiederanlaufbereich der KDCFILE. Der Datensatz beschreibt, welche Änderungen in den Verwaltungsdaten als Folge der Transaktion vorgenommen wurden. Bei einem Warmstart wird er von openUTM benutzt, um die Wirkung der Transaktion nachzuvollziehen. Die Größe des Wiederanlaufbereichs bestimmt, in welchen Zeitabständen Änderungen der Verwaltungsdaten in den Bereich Verwaltungsdaten der KDCFILE übernommen werden müssen.

In einer UTM-F-Anwendung werden nur in solchen Transaktionen Datensätze für den Wiederanlauf geschrieben, in denen Passwörter geändert oder per dynamischer Konfigurierung Änderungen an den Verwaltungsdaten vorgenommen wurden.

Die Datensätze im Wiederanlaufbereich werden von openUTM zusammengefasst, d.h. auf einer UTM-Seite dieses Bereichs haben meist mehrere Datensätze Platz.

In der KDCDEF-Generierung legt man die Größe des Puffers und des Wiederanlaufbereichs fest mit:

```
MAX ...,RECBUF=( number,length )
```

number Größe des Wiederanlaufbereichs pro Prozess in der KDCFILE in UTM-Seiten

length Größe des Puffers pro Prozess im Hauptspeicher, Angabe in Byte

Parameter *length* einstellen

Mit dem Parameter *length* reservieren Sie für jeden Prozess im Hauptspeicher einen Speicherbereich, der *length* Byte groß ist. openUTM verwendet diesen Speicherbereich, um Änderungen an den Verwaltungsdaten zwischenzeitlich zu sichern, solange eine Transaktion offen und damit rücksetzbar ist.

Für *length* muss der Platzbedarf von Transaktionen der Anwendung im Puffer an Hand von Standardwerten ermittelt werden:

- Bei einem Grundbedarf von 40 Byte pro Transaktion berücksichtigen Sie zusätzlich:
  - Bis zu 50 Byte pro KDCS-Aufruf, für MCOM-Aufruf aber 80 Byte.
    Auf 64 Bit Plattformen muss mit dem doppelten Speicherbedarf gerechnet werden.
  - Bis zu 300 Bytes pro ADMI-Aufruf für 32 und 64 Bit Plattformen.
- Bei verteilter Verarbeitung berücksichtigen Sie außerdem:
  - 300 Byte pro LU6.1-Kommunikationspartner
  - 200 Byte pro OSI TP-Partner
- Bei asynchroner Administration per FPUT-Aufruf ist zu beachten, dass alle Aufrufe FPUT NT aus einem Teilprogramm an den gleichen Administrations-TAC vom UTM-Administrationsprogramm in einer Transaktion bearbeitet werden. Dabei haben die einzelnen Administrationskommandos folgenden Platzbedarf im Puffer, den Sie für *length* berücksichtigen müssen:
  - 0 Byte pro Administrationskommando KDCHELP und KDCINF
  - für jedes andere Administrationskommando
    B
    X/W
    300 Byte unter BS2000/OSD
    360 Byte unter Unix- und Windows-Systemen

Wird RECBUF=*length* zu klein generiert, d.h. reicht der Puffer für eine Transaktion nicht aus, so lehnt openUTM den KDCS-Aufruf ab oder setzt die Transaktion zurückund beendet den Vorgang abnormal.

Parameter *number* einstellen

Mit dem Parameter *number* reservieren Sie für jeden Prozess einen Speicherbereich in der KDCFILE, der *number* UTM-Seiten groß ist. openUTM verwendet diesen Speicherbereich, um Änderungen an den Verwaltungsdaten abgeschlossener Transaktionen zwischenzeitlich zu sichern, bis die geänderten Verwaltungsdaten beim nächsten Periodic Write in die KDCFILE geschrieben werden. Auf einer UTM-Seite haben meist mehrere Datensätze Platz, da sie im Allgemeinen nur wenig umfangreicher sind als die entsprechende Information im Puffer.
KDCDEF sorgt dafür, dass ein zu klein definierter Wert *number* automatisch auf den kleinstmöglichen Wert erhöht wird.

Die Verwaltungsdaten in der KDCFILE, zusammen mit den für den Wiederanlauf geschriebenen Datensätzen, repräsentieren immer den letzten gültigen Zustand der Anwendung. Rechtzeitig bevor ein Wiederanlaufbereich während einer laufenden Anwendung vollgeschrieben wird, stößt openUTM automatisch einen Update der Verwaltungsdaten in der KDCFILE an (Periodic Write). Dabei werden alle Seiten mit Verwaltungsdaten, in denen Änderungen erfolgt sind, parallel zu laufenden Transaktionen in die KDCFILE geschrieben. Dadurch werden alle bisher geschriebenen Datensätze in den Wiederanlaufbereichen bedeutungslos.

Bei größerem Wiederanlaufbereich wird der Update der Verwaltungsdaten in der KDCFILE bei laufender Anwendung seltener durchgeführt. Nach einem Abbruch der Anwendung ist jedoch beim Warmstart meist eine sehr große Zahl von Datensätzen aus den Wiederanlaufbereichen einzuarbeiten, der Warmstart dauert also länger. Bei kleinem Wiederanlaufbereich ist es umgekehrt: Häufige Updates der Verwaltungsdaten in der laufenden Anwendung verkürzen die Zeit für einen Warmstart.

number sollte mindestens so groß sein, dass der im Wiederanlaufbereich zur Verfügung stehende Bereich ein Vielfaches des mit dem Parameter *length* generierten Puffers beträgt.

In einer UTM-F-Anwendung werden weit weniger Verwaltungsdaten in die KDCFILE zurückgeschrieben, beispielsweise geänderte Benutzerpasswörter und Generierungsdaten, die per dynamischer Konfigurierung geändert wurden. Der Wert *number* kann daher kleiner gewählt werden.

2.3.4 Neue KDCFILE im laufenden Betrieb erzeugen

Um bei einer Neu-Generierung die Ausfallzeit für eine UTM-Anwendung zu minimieren, ist es auch während des Betriebs einer UTM-Anwendung möglich, eine neue KDCFILE für diese Anwendung zu erzeugen. Dabei ist jedoch Folgendes zu beachten:

BS2000/OSD:

Der gesamte Basisname der neuen KDCFILE bestehend aus catid, userid und Präfix darf nicht mit dem der alten (aktuellen) KDCFILE übereinstimmen (zum Aufbau des Dateinamens siehe Seite 45).

Um das zu erreichen, gehen Sie wie folgt vor:

1. In der MAX-Anweisung geben Sie beim Parameter *filebase* den Dateinamen ohne Benutzerkennung *userid* an. Für *prefix* (siehe Seite 45) geben Sie den gleichen Wert an, wie bei der Generierung der „alten“ KDCFILE.
2. Starten Sie den KDCDEF-Lauf unter einer BS2000-Benutzerkennung, die von der, unter der die Anwendung abläuft, verschieden ist (z.B. userid2, wenn die alte KDCFILE :catid:$userid1.prefix.KDCA heißt).

Sie können dann zu einem späteren Zeitpunkt - außerhalb des Anwendungsbetriebs - die KDCFILE unter die Benutzerkennung *userid1* kopieren und wenn nötig einen KDCUPD-Lauf durchführen. Kopieren können Sie die KDCFILE z.B. mit:

```
/COPY-FILE FROM-FILE=$userid2.filebase.KDCA,TO-FILE=$userid1.filebase.KDCA
```

Starten Sie den KDCDEF-Lauf unter der Benutzerkennung *userid1* oder geben Sie in MAX KDCFILE= den Basisnamen mit Benutzerkennung an, dann bricht KDCDEF den KDCDEF-Lauf mit der Meldung K404 „DMS error D5B1 on file ...“ ab.

X/W *Unix- und Windows-Systeme:*

X/W Das Dateiverzeichnis, in das die neue KDCFILE geschrieben wird, darf nicht mit dem Basisverzeichnis der laufenden UTM-Anwendung übereinstimmen.

X/W Um das zu erreichen, gehen Sie wie folgt vor:

X/W 1. In der MAX-Anweisung geben Sie das Basisverzeichnis *filebase* mit „.“ an, d.h. die KDCFILE wird in das Dateiverzeichnis geschrieben, in dem der KDCDEF gestartet wird:

X/W
```
MAX KDCFILE=(.,S) oder MAX KDCFILE(.,D)
```

X/W 2. Sie starten den KDCDEF in einem anderen Dateiverzeichnis als dem Basisverzeichnis der UTM-Anwendung.

X/W Sie können dann zu einem späteren Zeitpunkt (außerhalb des Anwendungsbetriebs) die KDCFILE in das Basisverzeichnis kopieren und wenn nötig einen KDCUPD-Lauf durchführen.

X/W Starten Sie den KDCDEF-Lauf im Basisverzeichnis oder geben Sie in MAX KDCFILE= das Basisverzeichnis vollqualifiziert an, dann bricht KDCDEF den KDCDEF-Lauf mit der Meldung U185 ab.

2.4 Performance-Tuning

Die Zugriffe von openUTM auf die KDCFILE stellen insbesondere bei einer hohen Transaktionsrate einen wichtigen Faktor für die Performance einer UTM-Anwendung dar. Bei einer großen Konfiguration, d.h. großer KDCFILE, empfiehlt es sich, die Zugriffszeiten zu optimieren. Dazu stehen Ihnen folgende Möglichkeiten zur Verfügung:

- Aufspalten der KDCFILE (siehe unten)

X
- KDCFILE auf raw-device (nur auf Unix-Systemen, siehe Seite 58)

W
- KDCFILE auf Stripe Set (nur auf Windows-Systemen, siehe Seite 61)

B
B
In BS2000/OSD kann auch das HIPERFILE-Konzept zur Performance-Optimierung eingesetzt werden, siehe Handbuch „BS2000/OSD - Einführung in das DVS“.

2.4.1 KDCFILE aufspalten

Um das I/O-Verhalten Ihrer Anwendung zu verbessern, können Sie die KDCFILE aufspalten, indem Sie Pagepool und/oder Wiederanlaufbereich aus der KDCFILE auslagern. Eine Aufteilung von Pagepool und Wiederanlaufbereich auf mehrere Dateien ist vor allem bei höheren Transaktionsraten vorteilhaft, da openUTM seine Zugriffe auf diese Bereiche dann auf die verschiedenen Dateien verteilt.

Die Verwaltungsdaten stehen grundsätzlich in der Hauptdatei KDCA. Der ausgelagerte Pagepool und/oder der ausgelagerte Wiederanlaufbereich lassen sich per Generierung zusätzlich auf mehrere Dateien verteilen. Durch diese Aufteilung der KDCFILE auf mehrere physische Dateien werden, falls dies zur Nutzung mehrerer Hardwarepfade führt, die Zugriffszeiten verringert und die Performance erhöht.

Generierungshinweise

Mit folgenden Operanden der KDCDEF-Steueranweisung MAX legen Sie fest, welche Bereiche der KDCFILE bei der Generierung ausgelagert werden und wieviele Dateien für diese Bereiche jeweils erzeugt werden:

- Pagepool-Dateien

```
MAX...,PGPOOLFS=number
```

- Wiederanlaufbereichs-Dateien

```
MAX...,,RECBUFFS=number
```

Bei doppelter Dateiführung, die mit der Anweisung `MAX...,KDCFILE=(....,DOUBLE)` festgelegt wird, werden auch diese Dateien doppelt geführt.

Dateinamen

Die einzelnen Dateien der KDCFILE, die die ausgelagerten Bereiche enthalten, haben den selben Basisnamen *filebase* wie die Hauptdatei KDCA und besitzen die folgenden Namen:

– Pagepool-Dateien: P01A, P02A, P03A, ...
  Bei doppelt geführter KDCFILE werden zusätzlich die Dateien P01B, P02B, P03B, ... erzeugt.
– Wiederanlaufbereich: R01A, R02A, R03A, ...
  Bei doppelt geführter KDCFILE werden zusätzlich die Dateien R01B, R02B, R03B, ... erzeugt.

Beispiel

Sie wollen Ihre KDCFILE wie folgt einrichten:

– Den Pagepool auf 2 Dateien verteilen.
– Der Wiederanlaufbereich soll in einer eigenen Datei liegen.
– Die Dateien sollen doppelt geführt werden.

Für den Basisnamen wird im Beispiel der Platzhalter *FILEBASE* verwendet.

X/W
X/W
X/W

Unter Unix- oder Windows-Systemen ist *FILEBASE* das Dateiverzeichnis, in dem die Dateien abgelegt werden, und kann z.B. durch `/home/userutm/base` (Unix-Systeme) oder `C:\userutm\base` (Windows-Systeme) ersetzt werden.

Bei der KDCDEF-Generierung geben Sie dazu folgende MAX-Anweisung an:

```
MAX...,KDCFILE=(FILEBASE,DOUBLE),PGPOOLFS=2,RECBUFFS=1,...
```

KDCDEF erzeugt dann folgende Dateien:

		KDCFILE	Original	Kopie
B B B B B	BS2000/OSD:	Hauptdatei mit Verwaltungsdaten	*FILEBASE*.KDCA	*FILEBASE*.KDCB
		Pagepool	*FILEBASE*.P01A *FILEBASE*.P02A	*FILEBASE*.P01B *FILEBASE*.P02B
		Wiederanlaufbereich	*FILEBASE*.R01A	*FILEBASE*.R01B
X X X X X	Unix-Systeme:	Hauptdatei mit Verwaltungsdaten	*FILEBASE*/KDCA	*FILEBASE*/KDCB
		Pagepool	*FILEBASE*/P01A *FILEBASE*/P02A	*FILEBASE*/P01B *FILEBASE*/P02B
		Wiederanlaufbereich	*FILEBASE*/R01A	*FILEBASE*/R01B
W W W W W W	Windows-Systeme:	Hauptdatei mit Verwaltungsdaten	*FILEBASE*\KDCA	*FILEBASE*\KDCB
		Pagepool	*FILEBASE*\P01A *FILEBASE*\P02A	*FILEBASE*\P01B *FILEBASE*\P02B
		Wiederanlaufbereich	*FILEBASE*\R01A	*FILEBASE*\R01B

2.4.2 KDCFILE auf raw-device (Unix-Systeme)

X
X
X
X

Bei einer UTM-Anwendung auf Unix-Systemen werden Sie die Performance der Anwendung entscheidend verbessern, wenn Sie die KDCFILE auf raw-device betreiben, d.h. als zeichenorientierte Gerätedatei. Dazu legen Sie die KDCFILE auf eine eigene Plattenpartition, auf der kein Dateisystem angelegt ist.

X
X
X
X
X
X
X

Durch den direkten Zugriff auf die KDCFILE über eine Gerätedatei ohne Zwischenpufferung im Systemkern werden weniger Zeit und weniger Betriebsmittel benötigt als beim Zugriff über das Dateisystem, wenn die KDCFILE als normale Datei im Dateiverzeichnis *filebase* abgelegt wird. Auf der Plattenpartition wird die KDCFILE als zusammenhängender Datenbereich abgespeichert. Wird die KDCFILE als Datei innerhalb eines Dateisystems abgespeichert, dann werden die Daten der KDCFILE vom System häufig auf mehrere Speicherbereiche verteilt abgespeichert, was höhere Zugriffszeiten mit sich bringt.

X
X

Bei Aufspalten der KDCFILE auf mehrere Dateien (Auslagern von Pagepool und Wiederanlaufbereich) muss pro Datei eine **eigene** Plattenpartition verwendet werden.

Die raw-Partition des mit verwendeten Datenbanksystems sollte auf einer anderen Platte liegen.

Größe der benötigten Plattenpartition abschätzen

Damit der Systemverwalter eine hinreichend große Plattenpartition für Ihre KDCFILE anlegen kann, müssen Sie die Größe Ihrer KDCFILE ermitteln. Die Größe der KDCFILE hängt von den folgenden Faktoren ab:

- Anzahl der generierten Objekte, die über Namen angesprochen werden (Transaktionscodes, Benutzer, Teilprogramme, Clients und Drucker, Keysets, ferne Kommunikationspartner, Verbindungen für verteilte Verarbeitung etc.)
- generierte Größe des Pagepools (siehe Seite 48)
- generierte Größe des Wiederanlaufbereichs (siehe Seite 51)
- Anzahl der Workprozesse

Um die Größe der für Ihre KDCFILE benötigten Partition zu bestimmen, erzeugen Sie die KDCFILE mit Hilfe von KDCDEF als Datei *filebase*/KDCA. Das Kommando *ls -l* liefert Ihnen die Größe Ihrer KDCFILE. Berücksichtigen Sie, dass sich die Größe der KDCFILE durch spätere Änderungen der Konfiguration i.A. ändern wird. Wählen Sie die Plattenpartition deshalb vorsorglich entsprechend größer.

Raw-Gerätedatei einrichten

Der Systemverwalter legt die Platteneinteilung bei der Installation des Unix-Systems fest. Er sollte deshalb vor der Installation des Systems wissen, dass Sie Plattenpartitionen als raw-device ohne Dateisysteme für Ihre UTM-Anwendung benötigen. Der Systemverwalter kann dann bei der Installation mehrere kleinere Partitionen anlegen, die er je nach Speicherbedarf als Speicherbereich für Ihre KDCFILE zusammenfassen kann.

VORSICHT!
Viele Platten besitzen Verwaltungsinformation auf der ersten Spur. Dieser Bereich der Platte darf nicht zur Partition für die KDCFILE gehören.

Erzeugen Sie **kein** Dateisystem auf der Plattenpartition, auf der die KDCFILE abgespeichert werden soll. Hängen Sie die Plattenpartition **nicht** mit dem *mount*-Kommando ein.

Bitten Sie Ihren Systemverwalter, eine Gerätedatei für den Zugriff auf die Plattenpartition anzulegen. Stellen Sie sicher, dass der Zugriff auf die KDCFILE über eine zeichenorientierte Gerätedatei (raw-device) erfolgt, d.h. der Name der Gerätedatei muss mit *r* beginnen und das Identifikationszeichen muss *c* sein (erstes Zeichen der Ausgabe des Kommandos *ls -l*).

Eigentümer der Gerätedatei muss die Benutzerkennung sein, unter der die Anwendung abläuft. Die Gerätedatei darf nur für den Eigentümer lesend und schreibend benutzbar sein (Zugriffsrechte 600). Seien Sie bei der Vergabe der Zugriffsrechte auf die KDCFILE vorsichtig. Sie sind der einzige Schutz Ihrer KDCFILE vor unberechtigtem Zugriff.

Das Kommando *ls -l* auf die Gerätedatei

```
ls -l /dev/rxxxx
```

liefert dann die Ausgabe:

```
crw-------   1 utmaw    other      0,1030  Jul 14 15:13 /dev/rxxxx
```

KDCFILE auf die Gerätedatei schreiben

Es gibt zwei Möglichkeiten, wie Sie die KDCFILE auf die Plattenpartition legen können.

- Sie löschen die KDCFILE, die Sie zur Bestimmung der Dateigröße erzeugt haben. Erzeugen Sie dann mit dem *ln*-Kommando einen symbolischen Verweis zwischen Gerätedatei und *filebase*/KDCA. Erzeugen Sie anschließend die KDCFILE Ihrer Anwendung mit dem Generierungstool KDCDEF neu. openUTM schreibt die KDCFILE direkt auf die Gerätedatei.

  rm *filebase*/KDCA
  ln -s /dev/rxxxx *filebase*/KDCA
  utmpfad/ex/kdcdef

- Sie kopieren die bereits (für die Bestimmung der Größe) erzeugte KDCFILE mit dem *cp*- oder dem *dd*-Kommando auf die Gerätedatei und löschen dann die KDCFILE *filebase*/KDCA. Danach verknüpfen Sie die Gerätedatei mit *filebase*/KDCA mit Hilfe eines symbolischen Verweises (*ln*-Kommando).

  cp *filebase*/KDCA /dev/rxxxx
  rm *filebase*/KDCA
  ln -s /dev/rxxxx *filebase*/KDCA

Überprüfen Sie in beiden Fällen nach dem Absetzen des *ln*-Kommandos mit dem Kommando *ls -l*, ob die Verknüpfung *filebase*/KDCA mit der Gerätedatei existiert:

```
ls -l /dev/rxxxx filebase/KDCA
```

Ausgabe:

```
crw-------   1 utmaw   other  0,1030  Jul 14 15:13 /dev/rxxxx
lrwxrwxrwx   1 utmaw   other       9  Jul 14 15:49 filebase/KDCA -> /dev/rxxxx
```

X **Doppelte Dateiführung**

X Wenn Sie die KDCFILE aus Sicherheitsgründen doppelt führen, benötigen Sie zwei Plattenpartitionen. Die Plattenpartitionen sollten auf verschiedenen Platten liegen. Im Idealfall sollen verschiedene Controller die Platten bedienen. Der Systemverwalter muss für jede KDCFILE eine raw-Gerätedatei anlegen.

X Damit openUTM jede KDCFILE auf die dafür angelegte Gerätedatei schreiben kann, müssen Sie die folgenden symbolischen Verweise erzeugen:

```
X ln -s /dev/rxxx1 filebase/KDCA
X ln -s /dev/rxxx2 filebase/KDCB
```

2.4.3 KDCFILE auf Stripe Set (Windows-Systeme)

W Sie legen das Dateiverzeichnis *filebase* auf einem Stripe Set an (Windows Software RAID Level 0). Bei einem Stripe Set werden gleichgroße freie Bereiche auf verschiedenen Festplatten zu einem logischen Datenträger zusammengefasst. Die Daten einer KDCFILE auf einem Stripe Set werden also auf verschiedene Festplatten verteilt. Das hat schnellere Zugriffe und somit einen Performancegewinn der UTM-Anwendung zur Folge.

W Um eine höhere Datensicherheit zu erreichen, müssen Sie Stripe Sets mit Parität (RAID Level 5) verwenden. Stripe Sets mit Parität können jedoch nur unter Windows Server verwendet werden.

W Weitere Informationen zu Stripe Sets entnehmen Sie bitte der Dokumentation zu Windows-Systemen.

3 Hinweise zur Generierung einer UTM-Cluster-Anwendung

Eine UTM-Cluster-Anwendung ist, im Gegensatz zu einer stand-alone Anwendung, für den Ablauf auf mehr als einem Rechner vorgesehen. Der Verbund dieser Rechner wird Cluster genannt, die einzelnen Rechner, auf denen die Anwendung zum Ablauf kommen soll, werden Knoten genannt. Eine UTM-Cluster-Anwendung besteht aus mehreren identisch generierten UTM-Anwendungen, den Knoten-Anwendungen, die auf den einzelnen Knoten laufen.

Die Konfiguration der Anwendung einschließlich der KDCFILEs für alle Knoten wird in einem gemeinsamen Generierungslauf erstellt und ist daher grundsätzlich gleich.

Eine UTM-Cluster-Anwendung kann in BS2000/OSD auf bis zu 16 Knoten und in Unix- oder Windows-Systemen auf bis zu 32 Knoten verteilt werden.

Die Rechner eines Clusters müssen den gleichen Stand bezüglich Hardware- und Software-Konfiguration haben. Abweichungen mit kompatiblen Korrekturständen und Updates sind möglich. Eine Mischkonfiguration, z.B. BS2000- und Unix-Rechner, ist nicht möglich.

VORSICHT!
Die Knoten eines Clusters müssen grundsätzlich die gleiche Systemzeit haben.

Detaillierte Informationen zum Betrieb und insbesondere zur Änderungsgenerierung von UTM-Cluster-Anwendungen entnehmen Sie folgenden Handbüchern:

B
B
X/W
X/W

- openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“
- openUTM-Handbuch „Meldungen, Test und Diagnose in Unix- und Windows-Systemen“

3.1 Generierung einer UTM-Cluster-Anwendung

Die Generierung einer UTM-Cluster-Anwendung erfolgt in einem Generierungslauf, in dem das Dienstprogramm KDCDEF folgende Dateien erzeugt:

- die Cluster-Konfigurationsdatei
- die Cluster-User-Datei
- eine initiale KDCFILE
- und den Root-Source

Die Cluster-Konfigurationsdatei und die Cluster-User-Datei müssen für eine UTM-Cluster-Anwendung normalerweise nur einmal erzeugt werden. Über nachfolgende Generierungsläufe können die KDCFILE und/oder der Root-Source geändert werden. Die Cluster-User-Datei kann durch nachfolgende Generierungsläufe modifiziert werden.

Bei Konfigurationsänderungen kann in einem späteren Generierungslauf beispielsweise eine neue initiale KDCFILE mit zusätzlichen Objekten erzeugt werden.

Die initiale KDCFILE muss nach dem Generierungslauf für jede Knoten-Anwendung kopiert werden.

Auf die Cluster-Konfigurationsdatei, die Cluster-User-Datei und die KDCFILEs aller Knoten-Anwendungen muss von allen Knoten-Anwendungen aus zugegriffen werden können, siehe auch openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“, Kapitel „UTM-Cluster-Anwendung unter BS2000/OSD“ und openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter Unix- und Windows-Systemen“, Kapitel „UTM-Cluster-Anwendung unter Unix-Systemen.

Bild 5 zeigt, welche Dateien erstellt werden, wenn Sie eine UTM-Cluster-Anwendung definieren.

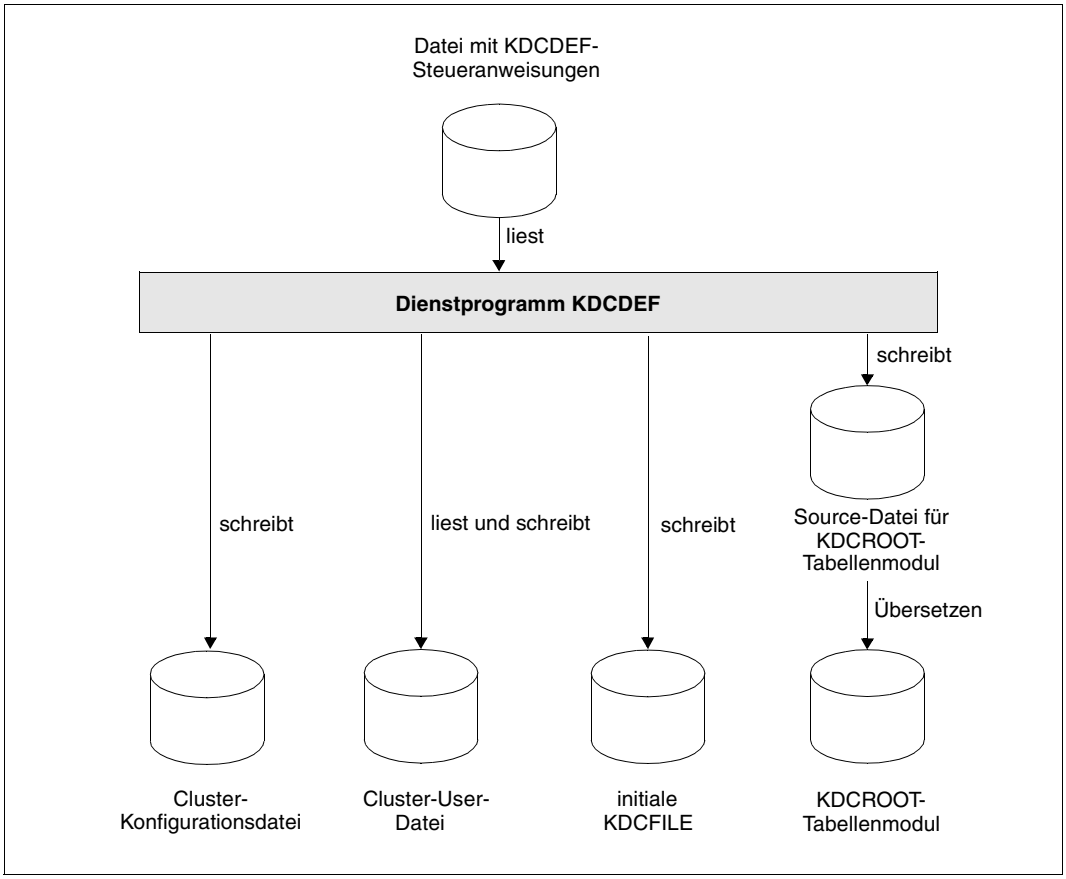


Bild 5: Ergebnis des KDCDEF-Laufs (bei OPTION ...,GEN=(KDCFILE, ROOTSRC, CLUSTER)) für eine UTM-Cluster-Anwendung

Wenn Sie bei der OPTION-Anweisung zusätzlich GEN=CLUSTER angeben, wird eine Cluster-Konfigurationsdatei erstellt.

Wenn die KDCDEF-Steueranweisungen eine CLUSTER-Anweisung enthalten, wird eine Cluster-User-Datei erstellt, oder eine bereits existierende Cluster-User-Datei ausgewertet und gegebenenfalls erweitert. Die Cluster-User-Datei enthält Angaben zur Verwaltung von Benutzerkennungen in einer UTM-Cluster-Anwendung.

Cluster-Konfigurationsdatei

Die Cluster-Konfigurationsdatei enthält Informationen zu allen Knoten-Anwendungen einer UTM-Cluster-Anwendung sowie Angaben zu Cluster-globalen Daten. Sie wird von allen Knoten-Anwendungen einer UTM-Cluster-Anwendung gemeinsam verwendet. Die Cluster-Konfigurationsdatei wird i.A. nur einmal für eine UTM-Cluster-Anwendung erstellt. Eine neue Cluster-Konfigurationsdatei können Sie nur einsetzen, nachdem alle Knoten-Anwendungen einer UTM-Cluster-Anwendung beendet wurden.

B Die Cluster-Konfigurationsdatei wird von KDCDEF im BS2000/OSD mit dem Dateinamen *cluster_filebase*.UTM-C.CFG angelegt. Dabei ist *cluster_filebase* der in der CLUSTER-Anweisung bei CLUSTER-FILEBASE angegebene Name. In BS2000/OSD kann die Cluster-Konfigurationsdatei umbenannt werden, wobei aber das Namens-Suffix .UTM-C.CFG erhalten bleiben muss.

X/W Auf Unix- und Windows-Systemen wird die Cluster-Konfigurationsdatei mit dem Dateinamen UTM-C.CFG in dem Dateiverzeichnis angelegt, das durch *cluster_filebase* bestimmt wurde. Für den Betrieb der UTM-Cluster-Anwendung kann die Cluster-Konfigurationsdatei umkopiert werden.

Beim Start einer Anwendung ist der gültige Name der Cluster-Filebase in den Startparametern anzugeben.

Cluster-User-Datei

Die Cluster-User-Datei dient zur Verwaltung von Benutzern einer UTM-Cluster-Anwendung.

Die Cluster-User-Datei wird i.A. nur einmal für eine UTM-Cluster-Anwendung erstellt. Die Datei kann während des Betriebs einer UTM-Cluster-Anwendung erweitert werden. Dies geschieht immer dann, wenn der Administrator neue Benutzer für eine UTM-Cluster-Anwendung definiert. Bei nachfolgenden Generierungsläufen zur Erstellung einer neuen KDCFILE müssen Sie deshalb immer auch die Cluster-User-Datei mit angeben. Die Einträge in der neuen KDCFILE werden mit den Einträgen in der bestehenden Cluster-User-Datei verknüpft, und die Cluster-User-Datei wird durch KDCDEF gegebenenfalls um Einträge für neue User erweitert.

B Die Cluster-User-Datei wird von KDCDEF im BS2000/OSD mit dem Dateinamen *cluster_filebase*.UTM-C.USER angelegt. Dabei ist *cluster_filebase* der in der CLUSTER-Anweisung bei CLUSTER-FILEBASE angegebene Name. In BS2000/OSD kann die Cluster-User-Datei umbenannt werden, wobei aber das Namens-Suffix .UTM-C.USER erhalten bleiben muss.

X/W Auf Unix- und Windows-Systemen wird die Cluster-User-Datei mit dem Dateinamen UTM-C.USER in dem Dateiverzeichnis angelegt, das durch *cluster_filebase* bestimmt wurde. Für den Betrieb der UTM-Cluster-Anwendung kann die Cluster-Konfigurationsdatei umkopiert werden.

Beim Start einer Anwendung ist der gültige Name der Cluster-Filebase in den Startparametern anzugeben.

KDCDEF-Anweisungen

Für die Generierung der UTM-Cluster-Anwendung sind spezifische Generierungsanweisungen nötig:

- Globale Eigenschaften einer UTM-Cluster-Anwendung definieren Sie mit der CLUSTER-Anweisung, siehe Seite 297. Dazu gehören z.B.
  - die Cluster-Filebase
  - der BCAMAPPL-Name und LISTENER-PORT für die Cluster-interne Kommunikation
  - Timer für die Überwachung
  - ein Failure- und ein Emergency-Kommando zum Aufruf bei Ausfall eines Knotens
- Knoten-spezifische Eigenschaften legen Sie für jede Knoten-Anwendung fest mit der CLUSTER-NODE-Anweisung, siehe Seite 306. Dazu gehören z.B.
  - der Basisname der KDCFILE, der Benutzer-Protokolldatei und der System-Protokolldatei SYSLOG
  - der Rechnername des Knotens

  Für jede Knoten-Anwendung müssen Sie eine eigene CLUSTER-NODE-Anweisung angeben.

- Eine neue Cluster-Konfigurationsdatei muss immer dann erstellt und eingesetzt werden, wenn Angaben bei der CLUSTER-Anweisung oder den CLUSTER-NODE-Anweisungen geändert wurden.
- Wenn bei der Generierung die Cluster-Konfigurationsdatei erzeugt werden soll, müssen Sie in der OPTION-Anweisung den Parameter GEN=CLUSTER angeben (siehe auch Seite 420).
- Für UTM-Cluster-Anwendungen müssen Sie bei der KDCDEF-Generierung MAX BLKSIZE=4K angeben. Für Anwendungen auf 64 Bit-Unix-Systemen wird dieser Wert implizit generiert.

Initiale KDCFILE

Die initiale KDCFILE wird beim Generierungslauf wie bei einer stand-alone Anwendung unter dem Basisnamen abgelegt, den Sie im Operanden KDCFILE der Anweisung MAX angeben.

Für den Anwendungslauf einer Knoten-Anwendung verwendet jede Knoten-Anwendung eine Kopie der initialen KDCFILE. Dazu müssen Sie nach dem Generierungslauf die initiale KDCFILE für jede Knoten-Anwendung einmal kopieren.

Da jede Knoten-Anwendung jeweils von einer anderen Knoten-Anwendung überwacht wird, müssen alle Knoten-Anwendungen gegenseitig auf alle KDCFILEs zugreifen können.

B
B
X/W
X/W

Informationen zum Starten und zur Überwachung der Knoten-Anwendungen und Ausfallerkennung entnehmen Sie folgenden Handbüchern:

- openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“
- openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter Unix- und Windows-Systemen“

3.2 Generierung einer Reserve-Knoten-Anwendung

Sie haben die Möglichkeit, bei der Generierung mit KDCDEF Reserve-Knoten-Anwendungen mit vorläufigen Werten anzulegen. Deren Rechnernamen und den Basisnamen der KDCFILE der Knoten-Anwendung können Sie später per Administration ändern. Dabei darf diese Knoten-Anwendung nicht aktiv sein.

- Legen Sie dazu mit der CLUSTER-NODE-Anweisung den vorläufigen, Knoten-spezifischen Basisnamen der KDCFILE sowie den Rechnernamen des Reserveknotens fest, siehe Seite 306.
- Später ändern Sie per Administration mit KC_MODIFY_OBJECT die Knoten-spezifischen Eigenschaften der Reserve-Knoten-Anwendung: Geben Sie den Objekttyp KC_CLUSTER_NODE an, um der auf Vorrat generierten Knoten-Anwendung tatsächliche Werte für den Rechnernamen des Cluster-Knotens und den Basisnamen der KDCFILE der Knoten-Anwendung zuzuordnen.

Weitere Informationen zu Einsatzmöglichkeiten für Reserve-Knoten-Anwendungen entnehmen Sie dem openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“ bzw. dem openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter Unix- und Windows-Systemen“.

Detaillierte Informationen zur Änderung der Knoten-spezifischen Eigenschaften per Administration entnehmen Sie dem openUTM-Handbuch „Anwendungen administrieren“.

3.3 Nutzung globaler Speicherbereiche

GSSB und ULS

Mit openUTM V6.0 ist in UTM-Cluster-Anwendungen die Verwendung der UTM-Speicherbereiche GSSB und ULS standardmäßig nicht erlaubt (GLOBAL-UTM-DATA=NO), weil mit openUTM eine Knoten-übergreifende Wirkung nicht möglich ist.

Eine Knoten-lokale Nutzung können Sie aber per Generierung erlauben, d.h. in jedem Cluster-Knoten existiert also eine eigene Ausgabe des jeweiligen Speicherbereichs. Wenn Sie diese Speicherbereiche Knoten-lokal verwenden wollen, müssen Sie bei der CLUSTER-Anweisung GLOBAL-UTM-DATA=YES angeben (siehe auch Seite 304).

In UTM-Cluster-Anwendungen, die mit CLUSTER GLOBAL-UTM-DATA=NO (Standard) generiert sind,

- müssen Sie bei der MAX-Anweisung GSSBS=0 angeben (siehe auch Seite 381).
- dürfen Sie keine ULS-Bereiche generieren (siehe auch Seite 537).

TLS

Die UTM-Speicherbereiche TLS sind in UTM-Cluster-Anwendungen Knoten-lokal angelegt, da jeder TLS einem LTERM oder (OSI-)LPAP zugeordnet ist, und zu einem Zeitpunkt zu jedem LTERM oder (OSI-)LPAP in jedem Cluster-Knoten eine Verbindung aufgebaut sein kann. In jedem Cluster-Knoten existiert also eine eigene Ausgabe des jeweiligen Speicherbereichs.

3.4 Verwendung von Benutzern mit RESTART=YES

In UTM V6.0 werden die Benutzer-spezifischen Kontexte Knoten-lokal verwaltet. In UTM-Cluster-Anwendungen ist daher die Verwendung des Vorgangswiederanlaufs standardmäßig nicht erlaubt, da in UTM V6.0 ein Vorgangswiederanlauf in anderen Knoten-Anwendungen nicht möglich ist. Bitte beachten Sie, dass LSSBs zum Vorgangskontext gehören und damit auch nur Knoten-lokal verwendet werden können. USER müssen daher mit RESTART=NO (Standard für UTM-Cluster-Anwendungen) generiert sein.

Einen Knoten-lokalen Vorgangswiederanlauf können Sie aber per Generierung erlauben, d.h. in jeder Knoten-Anwendung existiert dann eine eigene Ausgabe des Kontextes eines Benutzers. Ein Vorgangswiederanlauf findet somit immer mit dem Vorgangskontext statt, der in dieser Knoten-Anwendung zuletzt erzeugt wurde.

Wenn Sie den Knoten-lokalen Wiederanlauf verwenden wollen, müssen Sie bei der CLUSTER-Anweisung USER-RESTART=YES angeben. In diesem Fall

- können Sie USER RESTART=YES auch in UTM-Cluster-Anwendungen angeben. Die Benutzer-Kontexte sind dann Knoten-lokal.
- können Sie in UTM-Cluster-Anwendungen ohne USER die Angabe von RESTART=YES in der LTERM-Anweisung verwenden.

3.5 Besonderheiten in BS2000/OSD

B B B Wenn der Standard-Katalog der Benutzerkennung, unter der die UTM-Cluster-Anwendung gestartet werden soll, nicht der CATID des Shared Pubset entspricht, auf dem die KDCFILE abgelegt werden soll, müssen Sie die CATID in der Generierung angeben.

B B B **CLUSTER-NODE-Anweisung** auf Seite 306
Mit dem Operand CATID werden die CATIDs der globalen Cluster-Dateien festgelegt.

3.6 Besonderheiten auf Unix-/Windows-Systemen

X/W X/W X/W Die Verzeichnisse für die Ablage der globalen Cluster-Dateien müssen Sie vor dem Generierungslauf geeignet anlegen. Zum Generierungszeitpunkt müssen diese bereits vorhanden und zugreifbar sein.

X/W X/W Für UTM-Cluster-Anwendungen darf der Name bei MAX KDCFILE und CLUSTER-NODE FILEBASE maximal 27 Zeichen lang sein.

X/W X/W Für UTM-Cluster-Anwendungen auf 64 Bit Unix-Systemen dürfen Sie bei MAX KEYVALUE maximal den Wert 3900 angeben.

3.7 Besonderheiten bei LU6.1-Verbindungen

Für eine UTM-Cluster-Anwendung dürfen einem LPAP-Partner mehr Sessions (LSES-Anweisungen) zugeordnet werden als Verbindungen (CON-Anweisungen). Von KDCDEF wird dann als Warnung eine K438-Meldung ausgegeben, eine KDCFILE wird aber erzeugt.

Was Sie bei der LU6.1-Kommunikation einer stand-alone Anwendung mit einer UTM-Cluster-Anwendung beachten müssen, entnehmen Sie dem Abschnitt „LU6.1-LPAP-Bündel einer stand-alone Anwendung mit einer UTM-Cluster-Anwendung“ auf Seite 85.

4 Anwendungen für verteilte Verarbeitung generieren

Dieses Kapitel fasst die wichtigsten Generierungshinweise für Anwendungen mit verteilter Verarbeitung zusammen und beschreibt, wie die UTM-Generierung mit der Generierung des Transportsystems abgestimmt wird.

Unter verteilter Verarbeitung versteht man Server-Server-Kommunikation über die Protokolle LU6.1 und OSI TP. Über beide Protokolle kann mit System-übergreifender Transaktionssicherung gearbeitet werden. Das OSI TP-Protokoll erlaubt auch die Kommunikation mit OpenCPIC-Clients und LU6.2-Anwendungen. Die Generierung von OpenCPIC-Clients wird in Abschnitt „Clients an die Anwendung anschließen“ auf Seite 124f beschrieben. Weitere Informationen zur Kopplung mit LU6.2-Anwendungen finden Sie im openUTM-Handbuch „Verteilte Transaktionsverarbeitung zwischen openUTM- und CICS-, IMS- und LU6.2-Anwendungen“.

Die grundlegenden Prinzipien der verteilten Verarbeitung werden im openUTM-Handbuch „Konzepte und Funktionen“ dargestellt.

Um Anwendungen mit verteilter Verarbeitung zu generieren, müssen sowohl die Generierung der Einzel-Anwendungen fehlerfrei sein, als auch die Generierungsdaten aller beteiligten Anwendungen aufeinander abgestimmt werden. Da das Generierungstool KDCDEF immer nur die Generierungsdaten einer Anwendung auf Konsistenz und syntaktische Korrektheit prüfen kann, wirken sich Abstimmungsfehler im Allgemeinen erst beim Zusammenwirken der Anwendungen aus, beispielsweise beim Verbindungsaufbau.
Wenn WinAdmin zur Generierung verteilter UTM-Anwendungen genutzt wird, überprüft WinAdmin die Konsistenz der Anwendungs-übergreifenden Generierungsdaten. Generierungsfehler können damit weitgehend vermieden werden.

Generierung bei der Kopplung von stand-alone Anwendungen mit UTM-Cluster-Anwendungen

Generierungshinweise bei der Kopplung von stand-alone UTM-Anwendungen mit UTM-Cluster-Anwendungen finden Sie im Abschnitt „LU6.1-LPAP-Bündel einer stand-alone Anwendung mit einer UTM-Cluster-Anwendung“ auf Seite 85 sowie im Abschnitt „OSI-LPAP-Bündel“ auf Seite 100.

4.1 Verteilte Verarbeitung über das LU6.1-Protokoll

Zunächst werden einige für die Konfiguration relevante SNA-Begriffe im Zusammenhang erläutert, bevor auf die Generierung von UTM-Anwendungen mit verteilter Verarbeitung eingegangen wird.

Die SNA-Begriffe im nächsten Abschnitt sind *kursiv* gedruckt. Weitere Informationen zu den SNA-Begriffen finden Sie im openUTM-Handbuch „Verteilte Transaktionsverarbeitung zwischen openUTM- und CICS-, IMS- und LU6.2-Anwendungen“.

4.1.1 Transportverbindungen und SNA-Sessions

Die Kommunikation zwischen zwei Anwendungen erfolgt aus Sicht von openUTM über Transportverbindungen (im Sinne von TRANSDATA), über denen SNA-Sessions abgewickelt werden.

Die Sessions werden über *Sessionnamen* identifiziert. Die Sessionnamen dienen der Wiederherstellung einer unterbrochenen Kommunikation zwischen zwei Anwendungen. Wurde vor einer Störung mit einem Sessionnamen über eine der möglichen Transportverbindungen kommuniziert, so wird die Kommunikation nach Sessionwiederanlauf unter dem gleichen Sessionnamen, aber nicht unbedingt über die gleiche Transportverbindung fortgesetzt.

Eine Session wird mit der KDCDEF-Steueranweisung LSES definiert. Ihre Sessioneigenschaften werden mit der Steueranweisung SESCHA (Session Charakteristika) festgelegt, (z.B. wie die Session eröffnet, gesteuert und verwaltet wird).

Der Sessionname ist vergleichbar mit dem USER-Namen in UTM-Anwendungen: Ein USER kann ebenfalls einen unterbrochenen Vorgang an einer anderen Datenstation und damit über eine andere Transportverbindung fortsetzen. Damit die Sessionnamen in zwei miteinander verbundenen Anwendungen nicht gleich sein müssen, wird der Sessionname aus zwei Teilen zusammengesetzt (symbolisiert durch das ’+’-Zeichen):

sessionname = local-sessionname+remote-sessionname.

Jeder Teil des Sessionnamens ist maximal 8 Zeichen lang, d.h. der gemeinsame Sessionname hat max. 16 Bytes Länge. Der *local-sessionname* bezeichnet die gemeinsame Session in der lokalen Anwendung, der *remote-sessionname* die gleiche Session in der entfernten Anwendung. Daraus folgt, dass in der lokalen und der entfernten Anwendung die Sessionnamen der jeweils anderen Anwendung bekannt sein müssen. Der *local-sessionname* bildet in der lokalen Anwendung mit den dort definierten USER-Namen eine gemeinsame Namensmenge, der *remote-sessionname* bildet in der entfernten Anwendung mit den dort definierten USER-Namen eine andere gemeinsame Namensmenge.

Bei Vorgangsstart enthält das Feld "Benutzeridentifikation" im KB-Kopf einen lokalen Sessionnamen, wenn der Auftraggeber eine entfernte LU6.1-Anwendung ist.

Eine Session wird für die Dauer eines Dialogs zwischen einem Auftraggeber und Auftragnehmer exklusiv belegt (bracketing). D.h., dass ein anderer Auftraggeber:

– solange warten muss, bis die Session freigegeben ist oder

– seinen Auftrag später wiederholen muss, da er abgewiesen wird, oder

– eine andere freie Session belegt und seinen Auftrag startet. Voraussetzung dafür ist, dass zur entfernten Anwendung mehrere Transportverbindungen und mehrere Sessions existieren.

Eine der Partner-Anwendungen steuert den Auf- und Abbau der Session (SESCHA-Anweisung). Diese Anwendung wird *primary logical unit* oder abgekürzt *PLU* genannt. Die Initiative zum Aufbau der Session kann aber von beiden beteiligten Anwendungen ausgehen.

Beim Aufbau der Session einigen sich die Partner, welche Anwendung die Belegung der Session durch Aufträge steuert. Die Anwendung, die verwalten soll, wird *Contention Winner* genannt, die andere *Contention Loser*. Der Contention Winner kann, um einen Auftrag an den Partner zu geben, eine Session belegen, ohne vorher beim Contention Loser nachfragen zu müssen. Der Contention Loser muss erst beim Contention Winner nachfragen.

4.1.2 Generierungshinweise

Bei der Generierung von UTM-Anwendungen, die miteinander über das LU6.1-Protokoll kommunizieren wollen, ist Folgendes zu beachten.

1. In jeder Anwendung müssen für jede Partner-Anwendung eine oder zwei LPAP-Anweisungen mit dazugehörigen SESCHA-, CON- und LSES-Anweisungen generiert werden.

   Für eine Partner-Anwendung reicht eine LPAP-Anweisung, wenn nur von einer der beiden Anwendungen Aufträge gesendet werden. Sollen beide Anwendungen Aufträge an die jeweilige Partner-Anwendung senden, dann sollten in beiden Anwendungen jeweils zwei LPAP-Anweisungen generiert werden.

2. Ein LPAP, über das hauptsächlich Aufträge gesendet werden, wird in seiner zugeordneten SESCHA-Anweisung mit CONTWIN=NO generiert; damit wird die lokale Anwendung für dieses LPAP zum Contention Winner. Das entsprechende LPAP in der Partner-Anwendung muss dann mit CONTWIN=YES generiert werden.

3. Zu jedem LPAP muss eine gleiche Anzahl von CON- und LSES-Anweisungen generiert werden; die Anzahl der CON- bzw. LSES-Anweisungen bestimmt die Anzahl der parallelen Verbindungen, die über dieses LPAP mit der Partner-Anwendung möglich sind.

4. Für jede Verbindung bzw. jede Session muss in jeder der beiden Anwendungen jeweils eine CON- bzw. eine LSES-Anweisung generiert werden.

   Dabei müssen für jede CON-Anweisung die CON-Namen und die BCAMAPPL-Namen in der einen Anwendung mit denen in der Partner-Anwendung korrespondieren. Ebenso müssen für jede LSES-Anweisung die LSES-Namen und die RSES-Namen in der einen Anwendung mit denen in der Partner-Anwendung korrespondieren.

5. Alle CON-Anweisungen und alle LSES-Anweisungen eines LPAP müssen die gleiche Partner-Anwendung adressieren und sie müssen in der Partner-Anwendung genau einem LPAP-Namen zugeordnet sein. Es ist also verboten, für einen LPAP-Namen mehrere CON-Anweisungen zu generieren, die zu unterschiedlichen Anwendungen führen.

   Ebenso ist es verboten, für einen LPAP-Namen mehrere CON-Anweisungen zu generieren, die über ihre korrespondierenden CON-Anweisungen in der Partner-Anwendung dort unterschiedlichen LPAP-Anweisungen zugeordnet sind.

   Diese Generierungsfehler können von openUTM nicht erkannt werden, führen jedoch zu Fehlersituationen beim Verbindungs- bzw. Sessionaufbau sowie beim Sessionwiederanlauf.

6. Um mehrere parallele Verbindungen zwischen zwei Anwendungen aufbauen zu können, eröffnen die UTM-Anwendungen beim Transportsystem mehrere Transportsystem-Endpunkte. Jeder Transportsystem-Endpunkt einer UTM-Anwendung wird mittels einer eigenen BCAMAPPL-Anweisung generiert.

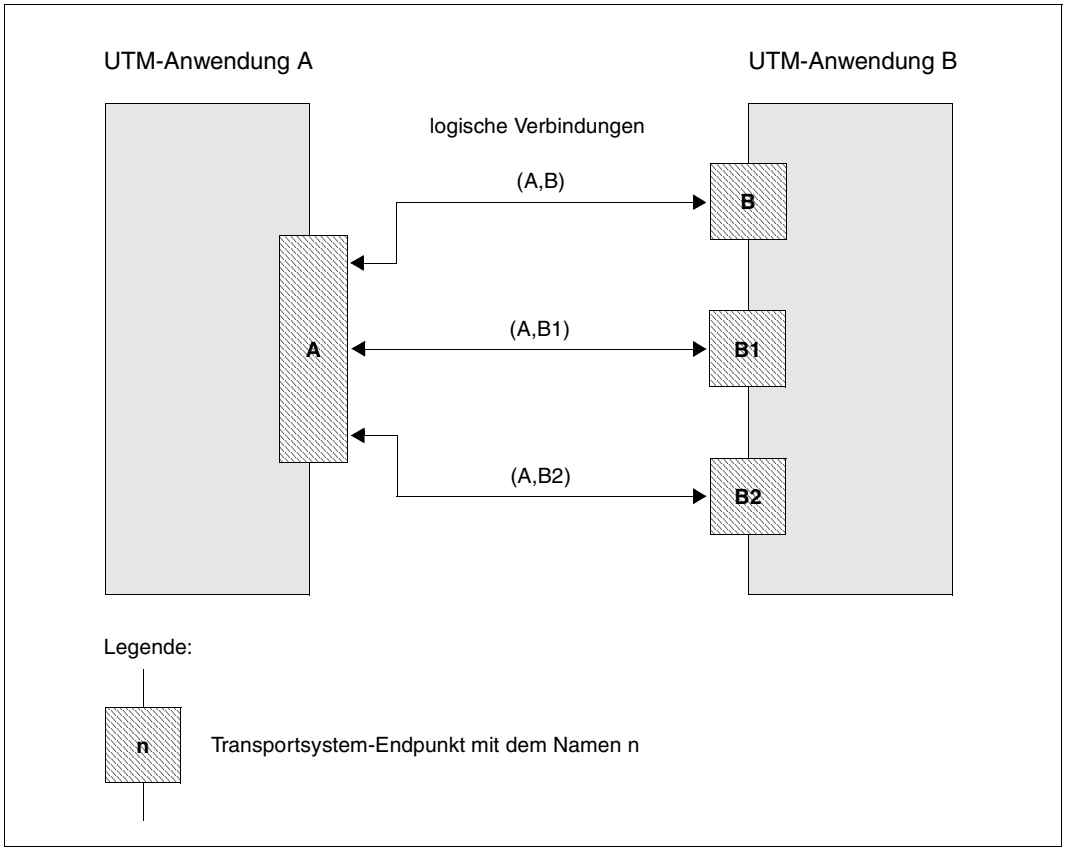


Bild 6: Zwei Anwendungen und mehrere Transportverbindungen

In obigem Beispiel entsprechen A bzw. B den Anwendungsnamen, die mit der Anweisung MAX APPLINAME= festgelegt wurden; B1 und B2 wurden mit eigenen BCAMAPPL-Anweisungen definiert.

Datenstationen können Verbindungen zur Anwendung A über den Anwendungsnamen A und Verbindungen zur Anwendung B über den Anwendungsnamen B aufbauen, während die Anwendung A Verbindungen zur Anwendung B über einen der Anwendungsnamen B, B1 oder B2 aufbauen kann.

B Aus Sicht der Netzadministration stellt sich damit die UTM-Anwendung B als mehrere
B BCAM-Anwendungen dar. BCAM-Administrationskommandos für einen der Anwendungs-
B namen B, B1 oder B2 wirken für die gesamte UTM-Anwendung B, d.h. das Kommando
B /BCLOSE B beendet die UTM-Anwendung B und meldet auch die Anwendungen B1 und
B B2 bei BCAM ab.

Zwischen je zwei Transportsystem-Endpunkten der beiden Anwendungen kann genau eine Transportverbindung aufgebaut werden. Sind also in der einen Anwendung zwei Transportsystem-Endpunkte generiert und in der anderen Anwendung drei, dann können zwischen den Anwendungen bis zu sechs parallele Verbindungen aufgebaut werden.

Werden für eine Partner-Anwendung sowohl ein Contention Winner LPAP als auch ein Contention Loser LPAP generiert (SESCHA-Anweisung), dann sollten Transportsystem-Endpunkte (BCAMAPPL-Anweisung), über die Contention Winner Verbindungen aufgebaut werden, nicht gleichzeitig zur Kommunikation für Contention Loser Verbindungen hergenommen werden! D.h. werden in einer Anwendung sowohl Contention Winner LPAPs als auch Contention Loser LPAPs generiert, dann sollten die BCAMAPPLs dieser Anwendung in zwei disjunkte Gruppen aufgeteilt werden, wobei die BCAMAPPLs der einen Gruppe nur Contention Winner Verbindungen zugeordnet werden und die BCAMAPPLs der anderen Gruppe entsprechend nur für Contention Loser Verbindungen verwendet werden.

4.1.3 Vorgehensweise bei der Generierung von LU6.1-Verbindungen

Bei der Generierung von zwei Anwendungen, die über das LU6.1-Protokoll miteinander kommunizieren möchten, sollte wie folgt vorgegangen werden.

1. LPAP- und SESCHA-Anweisungen

   Als erstes muss entschieden werden, ob jede der beiden Anwendungen in etwa gleich oft Aufträge an die andere Anwendung schickt, oder ob Aufträge überwiegend von einer der beiden Anwendungen gesendet werden.

   Im ersten Fall sollten in jeder der beiden Anwendungen jeweils zwei LPAP-Anweisungen generiert werden; im zweiten Fall wird in jeder der beiden Anwendungen nur jeweils eine LPAP-Anweisung benötigt. Dabei wird das LPAP, über das mehr Aufträge gesendet als empfangen werden, mit SESCHA ...,CONTWIN=NO generiert; das entsprechende LPAP in der Partner-Anwendung wird mit SESCHA ...,CONTWIN=YES generiert. Bei zwei LPAPs in einer Anwendung sollte ein LPAP mit
   SESCHA ... ,CONTWIN=NO und das andere mit
   SESCHA ...,CONTWIN=YES generiert werden.

   LPAP-Anweisung auf Seite 341
   Mit den folgenden Operanden können Sie einen LPAP-Partner als logischen Anschlusspunkt für die Partner-Anwendung definieren.

   - *lpapname*

     LPAP-Partnername, d.h. logischer Name der Partner-Anwendung, über den die Teilprogramme der lokalen Anwendung die Partner-Anwendung ansprechen. *lpapname* hat nur in der lokalen Anwendung eine Bedeutung.

- SESCHA=

  Die unter *sescha_name* in der SESCHA-Anweisung definierten Session-Eigenschaften für die Kommunikation zwischen der lokalen Anwendung und der Partner-Anwendung werden dem LPAP-Partner zugewiesen.

- PERMIT=

  legt die Berechtigungsstufe (Rechte zur Ausführung von Administrations- und Preselection-Funktionen) der Partner-Anwendung fest.

- QLEV=

  gibt die maximale Anzahl der Asynchron-Nachrichten an, die in der Message Queue des LPAP-Partners stehen dürfen.

- STATUS=

  definiert, ob mit der Partner-Anwendung sofort nach dem Start der lokalen Anwendung oder erst, nachdem der Status vom Administrator auf ON gesetzt wurde, zusammengearbeitet werden kann.

- BUNDLE=

  macht das LPAP zum Slave-LPAP eines LU6.1-LPAP-Bündels und gibt das zugehörige Master-LPAP an.

SESCHA-Anweisung auf Seite 474
Mit den folgenden Operanden können Sie Sessioneigenschaften definieren, die einem LPAP-Partner zugeordnet werden und damit der Partner-Anwendung, die sich über diesen LPAP-Partner anschließt.

- *sescha_name*

  definiert den Namen, unter dem die Sessioneigenschaften zusammengefasst werden. Dieser Name wird in der LPAP-Anweisung beim Operanden SESCHA= angegeben, um die Session Charakteristika einem LPAP-Partner zuzuordnen.

- CONTWIN=

  legt fest, ob die lokale Anwendung Contention Winner (NO) oder Contention Loser (YES) ist. Die Contention-Winner-Anwendung verwaltet die Session und regelt die Belegung der Session durch Aufträge.

  Standard: Bei PLU=N wird die lokale Anwendung Contention Loser, sonst Contention Winner.

- PLU=

  legt fest, welche Anwendung den Aufbau der Session initiiert, d.h. ob die Partner-Anwendung die primary logical unit (PLU) ist (YES) oder die lokale Anwendung (NO).
  Für eine der beteiligten Anwendungen muss PLU=Y angegeben werden und für die andere Anwendung PLU=N.

- CONNECT=

  legt fest, ob die lokale Anwendung beim Anwendungsstart die Verbindung zur Partner-Anwendung automatisch aufbauen soll (YES) oder ob die Verbindung zur Partner-Anwendung per Administrationskommando aufgebaut werden muss (NO).

Beispiel

Die Anwendung A sendet Aufträge an Anwendung B über das LPAP B1 und Anwendung B sendet Aufträge an Anwendung A über LPAP A2.

Anwendung A:

```
LPAP B1, SESCHA=B1
SESCHA B1, CONTWIN=NO, PLU=YES

LPAP B2, SESCHA=B2
SESCHA B2, CONTWIN=YES, PLU=NO
```

Anwendung B:

```
LPAP A1, SESCHA=A1
SESCHA A1, CONTWIN=YES, PLU=NO

LPAP A2, SESCHA=A2
SESCHA A2, CONTWIN=NO, PLU=YES
```

2. BCAMAPPL-Anweisungen

   Als nächstes wird festgelegt, wieviele parallele Verbindungen zwischen zwei LPAPs generiert werden sollen. Entsprechend dieser Anzahl werden in beiden Anwendungen mittels der BCAMAPPL-Anweisung zusätzliche Transportsystem-Endpunkte generiert. Zwischen jedem Transportsystem-Endpunkt der einen Anwendung und jedem Transportsystem-Endpunkt der anderen Anwendung kann genau eine Verbindung aufgebaut werden. Sollen z.B. neun parallele Verbindungen zwischen zwei LPAPs generiert werden, dann werden also auf jeder Seite mindestens jeweils drei BCAMAPPL-Anweisungen benötigt.

   Werden in einer Anwendung zwei LPAPs zu einer Partner-Anwendung generiert, dann sollten die BCAMAPPLs dieser Anwendung in zwei disjunkte Gruppen aufgeteilt werden, wobei das eine LPAP nur über BCAMAPPLs der einen Gruppe kommuniziert, während das zweite LPAP nur BCAMAPPLs der zweiten Gruppe verwendet.

BCAMAPPL-Anweisung auf Seite 289
Mit dem folgenden Operanden können Sie jeweils einen weiteren Anwendungsnamen für parallele Verbindungen zum Kommunikationspartner definieren.

- *appliname*

  Zusätzlicher (BCAM-)Name der UTM-Anwendung.

- T-PROT

  Gibt das Transportprotokoll an. Standard ist NEA.

Kommuniziert eine Anwendung mit mehreren Partner-Anwendungen, dann können die BCAMAPPLs, die für die Kommunikation mit einer Anwendung verwendet werden, auch für die Kommunikation zu den anderen Anwendungen eingesetzt werden.

3. CON- und LSES-Anweisungen

   Als nächstes werden jeder LPAP-Anweisung für jede parallele Verbindung über dieses LPAP jeweils eine CON- und eine LSES-Anweisung zugeordnet. Jede CON- und jede LSES-Anweisung muss dabei in jeder der beteiligten Anwendungen generiert werden und beide Generierungen müssen zueinander korrespondieren.

   Es gilt:

   – Jeder CON-Name in der einen Anwendung entspricht einem BCAMAPPL-Namen in der anderen Anwendung und umgekehrt.
   – Jeder LSES-Namen der einen Anwendung entspricht einem RSES-Namen in der anderen Anwendung und umgekehrt.

CON-Anweisung auf Seite 309
Mit den folgenden Operanden können Sie dem LPAP-Partner in der lokalen Anwendung die reale Partner-Anwendung zuordnen.

- *remote_appliname*

  Name der Partner-Anwendung, mit der über die logische Verbindung kommuniziert werden soll.

- BCAMAPPL=

  bezeichnet einen Namen der lokalen Anwendung, wie er in der Steueranweisung MAX oder BCAMAPPL festgelegt wurde. Es darf kein BCAMAPPL-Name angegeben werden, für den T-PROT=SOCKET generiert ist.

- LPAP=

  Name des LPAP-Partners der Partner-Anwendung, zu der die Verbindung aufgebaut werden soll. Der Name des LPAP-Partners, über den sich die Partner-Anwendung anschließt, muss mit der Anweisung LPAP *lpapname* definiert wer-

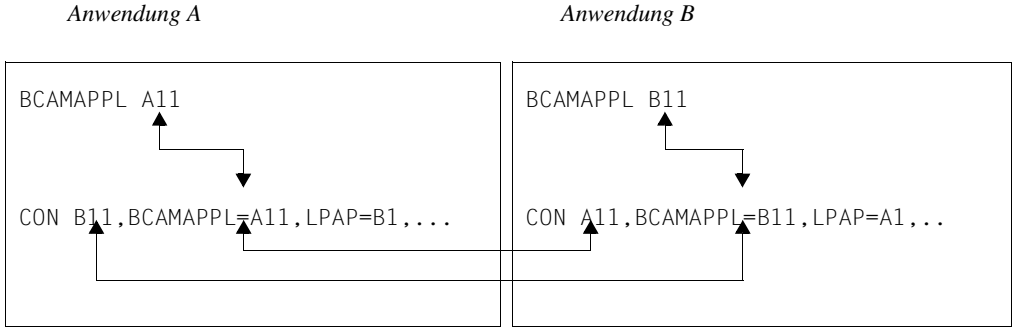
den.
Durch Angabe mehrerer CON-Anweisungen mit gleichem *lpapname* werden parallele Verbindungen zur Partner-Anwendung generiert.

- PRONAM=

  Name des Partner-Rechners.

Die CON-Anweisungen, mit denen in der lokalen Anwendung die Verbindung zur Partner-Anwendung und umgekehrt in der Partner-Anwendung die Verbindung zur lokalen Anwendung beschrieben werden, bezeichnen **eine** Verbindung. CON-Anweisungen müssen also immer paarweise auftreten. Bei parallelen Sessions werden für einen LPAP-Partner mehrere CON-Anweisungen generiert.

Im Beispiel wird eine Zuordnung zwischen dem LPAP-Partner B1 (generiert in Anwendung A) und dem LPAP-Partner A1 (generiert in Anwendung B) geschaffen:



LSES-Anweisung auf Seite 345
Mit den folgenden Operanden können Sie gemeinsame Sessionnamen für die Verbindung vereinbaren und dem LPAP-Partner zuordnen.

- *local_sessionname*

  Name der Session in der lokalen Anwendung.

- RSES=

  Name der Session in der Partner-Anwendung.

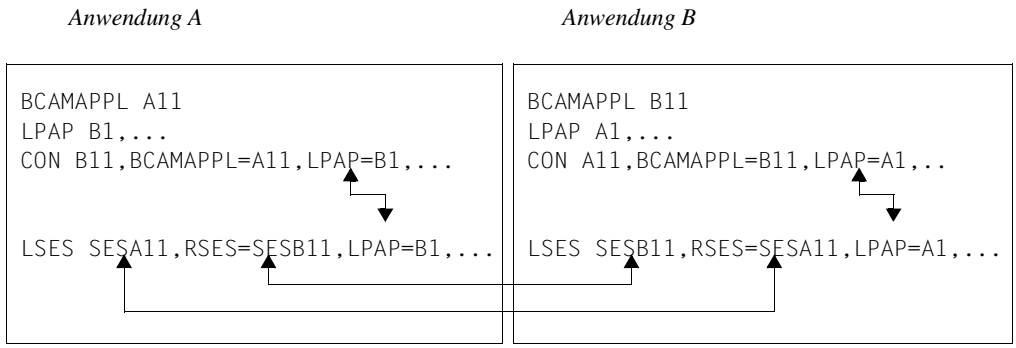
- LPAP=

  Name des LPAP-Partners, der der Partner-Anwendung zugeordnet wird. *local_sessionname* wird für die Kommunikation mit der Partner-Anwendung benutzt, die in der lokalen Anwendung dem LPAP-Partner *lpapname* zugeordnet ist.

Sessionnamen werden in der lokalen und der Partner-Anwendung vereinbart. LSES-Anweisungen müssen daher ebenfalls immer paarweise auftreten. Da der Sessionname den LPAP-Partnern zugeordnet ist, muss die Zuordnung der LPAP-Partner durch die LSES-Anweisung mit der Zuordnung der LPAP-Partner durch die CON-Anweisungen identisch sein.

Bei zwei einander zugeordneten LPAP-Partnern müssen die LSES- und RSES-Namen, die in den LSES-Anweisungen vereinbart wurden, zueinander passen (siehe folgendes Beispiel). Bei parallelen Sessions werden für einen LPAP-Partner *lpapname* mehrere LSES-Anweisungen mit unterschiedlichen Sessionnamen geschrieben.

Das vorige Beispiel wird nun wie folgt ergänzt.



Beispiel

Anwendung A:

```
BCAMAPPL A11
BCAMAPPL A12
LPAP B1, SESCHA=B1
SESCHA B1, CONTWIN=NO, PLU=YES

CON B11, BCAMAPPL=A11, LPAP=B1
CON B12, BCAMAPPL=A11, LPAP=B1
CON B11, BCAMAPPL=A12, LPAP=B1
CON B12, BCAMAPPL=A12, LPAP=B1
LSES SESA11, RSES=SESB11, LPAP=B1
LSES SESA12, RSES=SESB12, LPAP=B1
LSES SESA13, RSES=SESB13, LPAP=B1
LSES SESA14, RSES=SESB14, LPAP=B1

BCAMAPPL A21
BCAMAPPL A22
LPAP B2, SESCHA=B2
SESCHA B2, CONTWIN=YES, PLU=NO

CON B21, BCAMAPPL=A21, LPAP=B2
CON B22, BCAMAPPL=A21, LPAP=B2
CON B21, BCAMAPPL=A22, LPAP=B2
CON B22, BCAMAPPL=A22, LPAP=B2
LSES SESA21, RSES=SESB21, LPAP=B2
LSES SESA22, RSES=SESB22, LPAP=B2
LSES SESA23, RSES=SESB23, LPAP=B2
LSES SESA24, RSES=SESB24, LPAP=B2
```

Anwendung B:

```
BCAMAPPL B11
BCAMAPPL B12
LPAP A1, SESCHA=A1
SESCHA A1, CONTWIN=YES, PLU=NO

CON A11, BCAMAPPL=B11, LPAP=A1
CON A11, BCAMAPPL=B12, LPAP=A1
CON A12, BCAMAPPL=B11, LPAP=A1
CON A12, BCAMAPPL=B12, LPAP=A1
LSES SESB11, RSES=SESA11, LPAP=A1
LSES SESB12, RSES=SESA12, LPAP=A1
LSES SESB13, RSES=SESA13, LPAP=A1
LSES SESB14, RSES=SESA14, LPAP=A1

BCAMAPPL B21
BCAMAPPL B22
LPAP A2, SESCHA=A2
SESCHA A2, CONTWIN=NO, PLU=YES

CON A21, BCAMAPPL=B21, LPAP=A2
CON A21, BCAMAPPL=B22, LPAP=A2
CON A22, BCAMAPPL=B21, LPAP=A2
CON A22, BCAMAPPL=B22, LPAP=A2
LSES SESB21, RSES=SESA21, LPAP=A2
LSES SESB22, RSES=SESA22, LPAP=A2
LSES SESB23, RSES=SESA23, LPAP=A2
LSES SESB24, RSES=SESA24, LPAP=A2
```

Hinweise

- Bei Anwendungen mit verteilter Verarbeitung muss evtl. der Wert *length* der Anweisung MAX...,RECBUF=(*number*,*length*),... vergrößert werden. Hinweise, wie groß dieser Wert zu wählen ist, finden Sie im Abschnitt „Wiederanlaufbereich“ auf Seite 51.
- Das Verhalten einer Anwendung kann durch die Wahl der Timer (IDLETIME= in der SESCHA-Anweisung, CONCTIME und PTCTIME in der UTMD-Anweisung) beeinflusst werden.

4.1.4 LU6.1-LPAP-Bündel

LU6.1-LPAP-Bündel ermöglichen eine automatische Verteilung von Nachrichten auf mehrere LPAP-Partner. Soll eine UTM-Anwendung sehr viele Nachrichten mit einer Partner-Anwendung austauschen, kann es für die Lastverteilung sinnvoll sein, mehrere Instanzen der Partner-Anwendung zu starten und die Nachrichten auf die einzelnen Instanzen zu verteilen. In einem LU6.1-LPAP-Bündel übernimmt openUTM die Verteilung der Nachrichten an die Instanzen der Partner-Anwendung. Dazu müssen die Teilprogramme im APRO-Aufruf das MASTER-LU61-LPAP adressieren.

Ein Anwendungsfall für eine solche Verteilung der Nachrichten ist die Kommunikation einer UTM-Anwendung mit einer UTM-Cluster-Anwendung. Die Nachrichten an die UTM-Cluster-Anwendung können somit auf die einzelnen Knoten-Anwendungen verteilt werden. Detaillierte Information dazu finden Sie im Abschnitt „LU6.1-LPAP-Bündel einer stand-alone Anwendung mit einer UTM-Cluster-Anwendung“ auf Seite 85.

Ein LU6.1-LPAP-Bündel besteht aus einem Master-LPAP und mehreren Slave-LPAPs. Die Slave-LPAPs werden dem Master-LPAP bei der Generierung zugeordnet. Die einzelnen Slave-LPAPs adressieren dabei (im Normalfall) verschiedene Partner-Anwendungen.

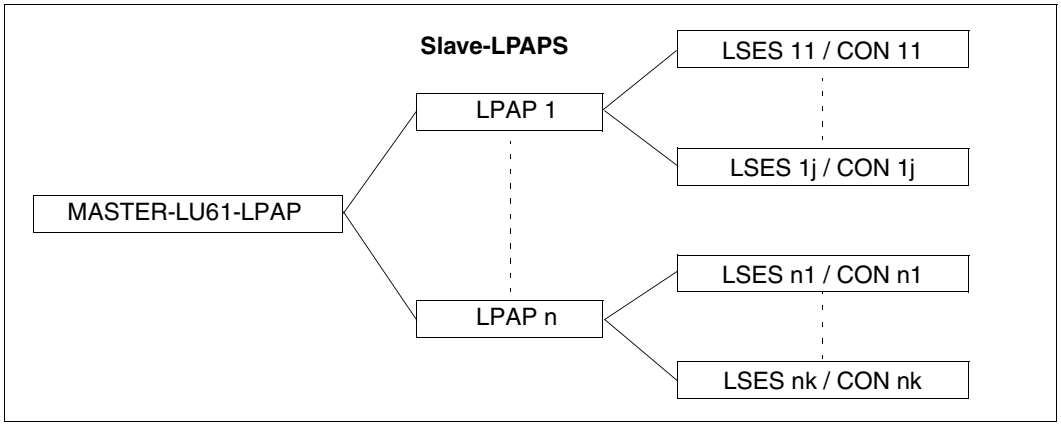


Bild 7: Beispiel LU6.1-LPAP-Bündel

LU6.1-LPAP-Bündel generieren

MASTER-LU61-LPAP-Anweisung auf Seite 366
Legt den Namen und die Eigenschaften des Master-LPAP in einem LU6.1-LPAP-Bündel fest:

- *master-lpap-name*

  Name für das Master-LPAP.

- STATUS=

  gibt an, ob Nachrichten an dieses LPAP-Bündel gesendet werden können.

LPAP-Anweisung auf Seite 341
Für die Generierung eines Slave-LPAP müssen die folgenden Eigenschaften festgelegt werden:

- *lpap-name*

  Name des Slave-LPAP.

- BUNDLE=master-lpap-name

  Name des Master-LPAP. Das Master-LPAP, der hier angegeben wird, muss in einer Anweisung MASTER-LU61-LPAP definiert sein. Durch Angabe von BUNDLE wird dieses LPAP zum Slave-LPAP des angegebenen Master-LPAP.

```
MASTER-LU61-LPAP master, ...
LPAP slave-lpap, BUNDLE=master, ...
```

CONs von LPAPs eines LU6.1-LPAP-Bündels

- Einem Master-LPAP dürfen keine physikalischen Verbindungen (CONs) zugeordnet werden. Es darf also nicht als LPAP in einer CON-Anweisung angegeben werden. Das Master-LPAP nutzt immer die Verbindungen, die den Slave-LPAPs zugeordnet sind.

Verteilung von Nachrichten

Details dazu entnehmen Sie dem Abschnitt „Verteilung von Nachrichten“ auf Seite 102.

Anzeige im KB-Kopf

Details dazu entnehmen Sie dem Abschnitt „Anzeige im KB-Kopf“ auf Seite 103.

4.1.5 LU6.1-LPAP-Bündel einer stand-alone Anwendung mit einer UTM-Cluster-Anwendung

Beachten Sie bei der Generierung der LU6.1-Kommunikation einer stand-alone Partner-Anwendung mit einer UTM-Cluster-Anwendung Folgendes:

- Eine Partner-Anwendung muss für jeden Knoten der UTM-Cluster-Anwendung, mit dem sie kommunizieren will, ein LPAP mit jeweils einer bestimmten Anzahl von Sessions und Verbindungen generieren.
- Zur Adressierung der UTM-Cluster-Anwendung sollte in der Partner-Anwendung ein LU6.1-LPAP-Bündel generiert werden, dessen Slave-LPAPs den Cluster-Knoten zugeordnet sind (siehe Abschnitt „MASTER-LU61-LPAP – Master-LPAP eines LU6.1-LPAP-Bündels definieren“ auf Seite 366).
- In der UTM-Cluster-Anwendung müssen für das LPAP, das die Partner-Anwendung repräsentiert, mehr Sessions (LSES) als Verbindungen (CON) generiert werden, und zwar für jede Verbindung eine Session pro Cluster-Knoten.
- Jeder Cluster-Knoten benötigt für das entsprechende LPAP nur genau die Anzahl von Verbindungen, die jedem LPAP in der Partner-Anwendung zugeordnet sind. Da jedoch alle Cluster-Knoten identische Generierungen haben, müssen in jedem Cluster-Knoten die Sessions aller LPAPs der Partner-Anwendung generiert werden.

Eine Verbindung einer auf diese Weise generierten LU6.1-Partner-Anwendung muss immer von der stand-alone Partner-Anwendung aufgebaut werden, nie von der UTM-Cluster-Anwendung. Generieren Sie in diesem Fall die LU6.1-Verbindungen in der UTM-Cluster-Anwendung nicht mit Autoconnect, d.h. geben Sie bei der SESCHA-Anweisung CONNECT=NO (Standardwert) an (siehe auch Seite 474).

Administrativ kann eine LU6.1-Session auch von der UTM-Cluster-Anwendung aus aufgebaut werden, allerdings nur mit dem KDCLSES-Kommando bzw. bei programmierter Administration über die Modifikation des Objekts LSES. Dazu muss der Administrator eine Session (LSES) auswählen, deren entfernte Session (RSES) in der Partner-Anwendung dem LPAP zugeordnet ist, das in der entfernten Anwendung die lokale Knoten-Anwendung repräsentiert.

Beispiel:

Das folgende Beispiel zeigt eine Generierung, bei der die stand-alone Anwendung SA auf dem Rechner HOSTSA mit der UTM-Cluster-Anwendung CA auf den Cluster-Knoten NODECAX, NODECAY und NODECAZ gekoppelt wird. Es werden jeweils 4 Verbindungen zwischen der stand-alone Anwendung und jeder Knoten-Anwendung generiert. Für die LPAPs, die die Knoten-Anwendungen in der stand-alone Anwendung repräsentieren, wird ein MASTER-LU61-LPAP generiert, das die UTM-Cluster-Anwendung repräsentiert.

Stand-alone Anwendung SA auf HOSTSA	UTM-Cluster-Anwendung CA auf NODECAX/Y/Z
`BCAMAPPL SA11`	`BCAMAPPL CA11`
`BCAMAPPL SA12`	`BCAMAPPL CA12`
`MASTER-LU61-LPAP MLPAPCA`	
`LPAP LPAPCAX,SESCHA=SESCHCA-` `  , BUNDLE=MLPAPCA`	`LPAP LPAPSA, SESCHA=SESCHSA`
`LPAP LPAPCAY,SESCHA=SESCHCA-` `  , BUNDLE=MLPAPCA`	
`LPAP LPAPCAZ,SESCHA=SESCHCA-` `  , BUNDLE=MLPAPCA`	
`SESCHA SESCHCA, CONTWIN=NO, PLU=YES`	`SESCHA SESCHSA, CONTWIN=YES, PLU=NO`
`CON CA11, PRONAM=NODECAX -` `  , BCAMAPPL=SA11, LPAP=LPAPCAX`	`CON SA11, PRONAM=HOSTSA -` `  , BCAMAPPL=CA11, LPAP=LPAPSA`
`CON CA12, PRONAM=NODECAX -` `  , BCAMAPPL=SA11, LPAP=LPAPCAX`	`CON SA11, PRONAM=HOSTSA -` `  , BCAMAPPL=CA12, LPAP=LPAPSA`
`CON CA11, PRONAM=NODECAX -` `  , BCAMAPPL=SA12, LPAP=LPAPCAX`	`CON SA12, PRONAM=HOSTSA -` `  , BCAMAPPL=CA11, LPAP=LPAPSA`
`CON CA12, PRONAM=NODECAX -` `  , BCAMAPPL=SA12, LPAP=LPAPCAX`	`CON SA12, PRONAM=HOSTSA -` `  , BCAMAPPL=CA12, LPAP=LPAPSA`
`CON CA11, PRONAM=NODECAY -` `  , BCAMAPPL=SA11, LPAP=LPAPCAY`	
`CON CA12, PRONAM=NODECAY -` `  , BCAMAPPL=SA11, LPAP=LPAPCAY`	
`CON CA11, PRONAM=NODECAY -` `  , BCAMAPPL=SA12, LPAP=LPAPCAY`	
`CON CA12, PRONAM=NODECAY -` `  , BCAMAPPL=SA12, LPAP=LPAPCAY`	
`CON CA11, PRONAM=NODECAZ -` `  , BCAMAPPL=SA11, LPAP=LPAPCAZ`	
`CON CA12, PRONAM=NODECAZ -` `  , BCAMAPPL=SA11, LPAP=LPAPCAZ`	
`CON CA11, PRONAM=NODECAZ -` `  , BCAMAPPL=SA12, LPAP=LPAPCAZ`	
`CON CA12, PRONAM=NODECAZ -` `  , BCAMAPPL=SA12, LPAP=LPAPCAZ`	

```
LSES SAA2CAX, RSES= CA12SA1-          LSES CA12SA1, RSES= SAA2CAX-
  , LPAP=LPAPCAX                        , LPAP=LPAPSA
LSES SAB2CAX, RSES= CA12SA2-          LSES CA12SA2, RSES= SAB2CAX-
  , LPAP=LPAPCAX                        , LPAP=LPAPSA
LSES SAC2CAX, RSES= CA12SA3-          LSES CA12SA3, RSES= SAC2CAX-
  , LPAP=LPAPCAX                        , LPAP=LPAPSA
LSES SAD2CAX, RSES= CA12SA4-          LSES CA12SA4, RSES= SAD2CAX-
  , LPAP=LPAPCAX                        , LPAP=LPAPSA

LSES SAA2CAY, RSES= CA22SA1-          LSES CA22SA1, RSES= SAA2CAY-
  , LPAP=LPAPCAY A                      , LPAP=LPAPS
LSES SAB2CAY, RSES= CA22SA2-          LSES CA22SA2, RSES= SAB2CAY-
  , LPAP=LPAPCAY                        , LPAP=LPAPSA
LSES SAC2CAY, RSES= CA22SA3-          LSES CA22SA3, RSES= SAC2CAY-
  , LPAP=LPAPCAY                        , LPAP=LPAPSA
LSES SAD2CAY, RSES= CA22SA4-          LSES CA22SA4, RSES= SAD2CAY-
  , LPAP=LPAPCAY                        , LPAP=LPAPSA

LSES SAA2CAZ, RSES= CA32SA1-          LSES CA32SA1, RSES= SAA2CAZ-
  , LPAP=LPAPCAZ                        , LPAP=LPAPSA
LSES SAB2CAZ, RSES= CA32SA2-          LSES CA32SA2, RSES= SAB2CAZ-
  , LPAP=LPAPCAZ                        , LPAP=LPAPSA
LSES SAC2CAZ, RSES= CA32SA3-          LSES CA32SA3, RSES= SAC2CAZ-
  , LPAP=LPAPCAZ                        , LPAP=LPAPSA
LSES SAD2CAZ, RSES= CA32SA4-          LSES CA32SA4, RSES= SAD2CAZ-
  , LPAP=LPAPCAZ                        , LPAP=LPAPSA
```

Wenn Sie in der Knoten-Anwendung CA auf Knoten NODECAY eine LU6.1-Session zur stand-alone Anwendung SA aufbauen wollen, kann dies nur über die Sessions CA22SA1, CA22SA2, CA22SA3 und CA22SA4 erfolgen, da deren entfernte Sessions SAA2CAY, SAB2CAY, SAC2CAY und SADA2CAY dem LPAP LPAPCAY zugeordnet sind.

Beispiel:

```
KDCLSES LSES=CA22SA1, ACT=C
```

4.2 Verteilte Verarbeitung über das OSI TP-Protokoll

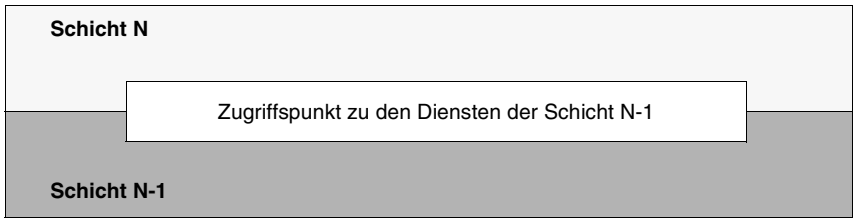
Bevor auf Regeln und Empfehlungen zur Generierung von UTM-Anwendungen für die verteilte Verarbeitung über OSI TP eingegangen wird, werden für die Generierung relevante OSI-Begriffe erläutert. Die OSI-Begriffe in diesem Abschnitt sind *kursiv* gedruckt.

4.2.1 OSI-Begriffe

Für OSI (Open System Interconnection) sind die Regeln und Leistungen normiert, die zwei Partner einhalten bzw. erbringen müssen, wenn sie miteinander kommunizieren wollen.

Die Organisation ISO (International Organization for Standardization) hat dazu das *OSI-Referenzmodell* entwickelt. Im *OSI-Referenzmodell* sind die Kommunikationsaufgaben auf *7 Schichten (Instanzen)* verteilt. Für jede Schicht ist festgeschrieben, welche Leistungen sie erbringen muss. Die 7 Schichten bauen hierarchisch aufeinander auf. Jeder Schicht stehen die Leistungen der darunterliegenden Schicht als *Dienst* zur Verfügung. Diese Dienste stehen der höheren Schicht an *Dienstzugriffspunkten* (Service Access Point) zur Verfügung.

Bei der Kommunikation greift die Anwendung über einen solchen Zugriffspunkt auf die Dienste des Kommunikationssystems zu:



Wollen nun zwei Anwendungen miteinander kommunizieren und Daten austauschen, so muss zwischen ihnen eine *Transportverbindung* bestehen. Eine Transportverbindung kann nur zwischen zwei *adressierbaren Einheiten* im Netz aufgebaut werden. Jede Anwendung muss also über eine im Netz eindeutige *Adresse* identifizierbar sein.

In der OSI-Welt wird jedem Dienstzugriffspunkt eine Adresse zugeordnet, nicht der Anwendung. Die Anwendung ist dann im Netz identifizierbar über die Adresse des Zugriffspunktes, über den die Anwendung die Kommunikation abwickelt.

Jedem Dienstzugriffspunkt wird deshalb eine im Netz eindeutige Adresse zugeordnet. Sie setzt sich zusammen aus einem *Selektor* und der Adresse des darunterliegenden Zugriffspunktes. Wie die Adressen der Zugriffspunkte im OSI-Referenzmodell gebildet werden, ist in der folgenden Grafik dargestellt.

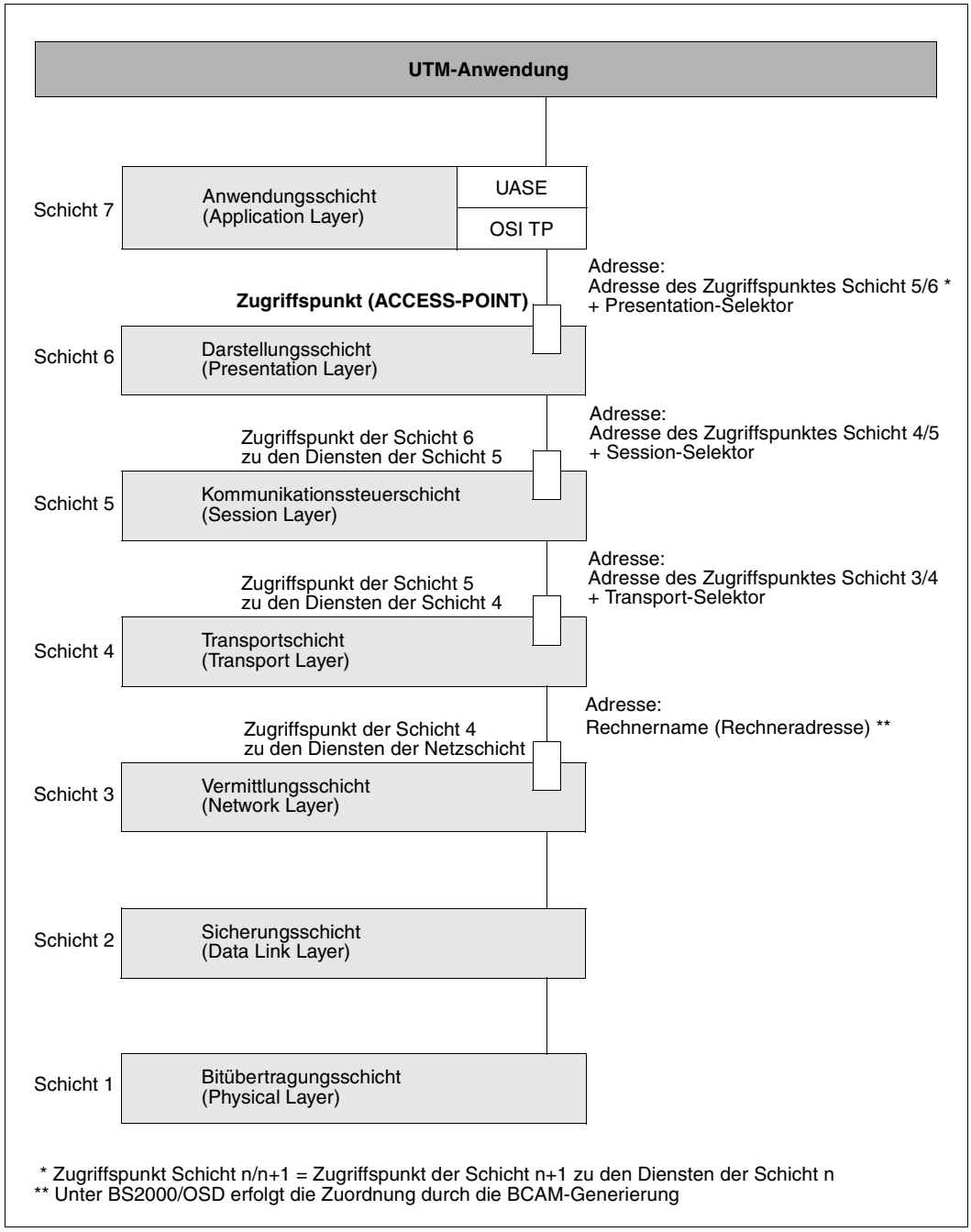


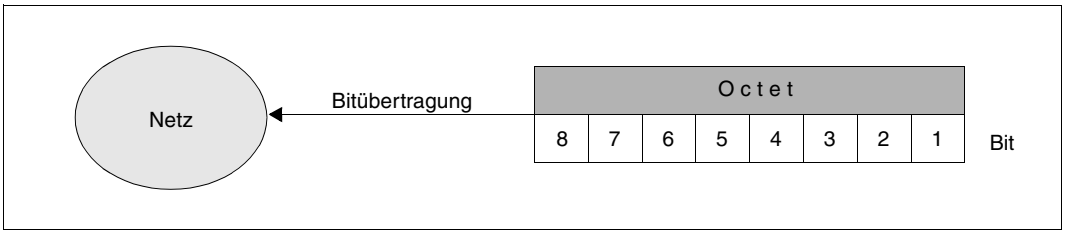
Bild 8: Adressen der Dienstzugriffspunkte im OSI-Referenzmodell

Eine UTM-Anwendung bindet sich an einen Dienstzugriffspunkt, wenn sie mit anderen Anwendungen im Netz kommunizieren will. Die Bindung wird mit der KDCDEF-Anweisung ACCESS-POINT generiert. Über die Adresse dieses Zugriffspunktes ist die UTM-Anwendung dann von ihren Partnern im Netz ansprechbar.

Der Aufbau der Adresse des Zugriffspunktes über den die Anwendung adressierbar ist, ist abhängig von der Hierarchie des Zugriffspunktes. Wickelt die UTM-Anwendung die Kommunikation über OSI TP ab, so bindet sie sich an einen Dienstzugriffspunkt zu den Diensten der Schicht 6. Die Adresse des Zugriffspunktes im lokalen System besteht dann aus dem *Transportselektor* (Selektor der *Transportschicht*), dem *Session-Selektor* (Selektor der *Kommunikationssteuerschicht*) und dem *Presentation-Selektor* (Selektor der *Darstellungsschicht*). Die Netzadresse des Zugriffspunktes besteht damit aus der Netzadresse des Rechners und der Adresse des Zugriffspunktes im lokalen System.

Die Selektoren der einzelnen Schichten müssen im lokalen System eindeutig sein. Die Rechneradresse ist im Netz eindeutig. Die Werte, die Sie für die Adresse des Zugriffspunktes angeben, müssen Sie mit dem Netzverwalter absprechen.

Die Selektoren setzen sich aus *Octets* zusammen. Octet ist eine andere Bezeichnung für ein Byte (8 Bit). Die Bezeichnung legt aber auch die Nummerierung der Bits in dem Byte fest und damit die Reihenfolge, in der die Bits übertragen werden.



Reihenfolge der Bitübertragung eines Octets

Zu einem Kommunikationspartner können mehrere *parallele Verbindungen* (auch *Association* genannt) bestehen. Alle Verbindungen zu einem fernen Partner werden in **einer** OSI-CON-Anweisung generiert. Die Anzahl der parallelen Verbindungen zu einem Partner legen Sie in einer OSI-LPAP-Anweisung fest. Jeder einzelnen Verbindung muss ein im lokalen System eindeutiger Name, der *Association-Name*, zugeordnet werden. Die Namen der Associations zu einem fernen Partner generieren Sie ebenfalls in der OSI-LPAP-Anweisung.

Jede Verbindung zwischen zwei Partnern wird von einem dieser Partner verwaltet. Der Partner, der die Verbindung verwaltet, heißt *Contention Winner*. Der andere Partner ist der *Contention Loser*. Aufträge können von beiden Partnern gestartet werden. Starten beide Partner zur gleichen Zeit einen Auftrag, so belegt der Auftrag des Contention Winners die Verbindung. Contention Winner einer Verbindung sollte immer der Kommunikationspartner sein, der am häufigsten Aufträge startet.

Existieren zwischen zwei Partnern mehrere parallele Verbindungen, so muss nicht für jede Verbindung festgelegt werden, welcher Partner Contention Loser und welcher Contention Winner ist. Es wird lediglich festgelegt, für wieviele der Verbindungen die einzelnen Partner Contention Winner sein sollen (Anweisung OSI-LPAP). Wiederum sollte der Partner die meisten Verbindungen verwalten, der am häufigsten Aufträge startet.

openUTM unterstützt den *TPSU-title* (**T**ransaction **P**rocessing **S**ervice **U**ser). Dies ist ein Benutzer der OSI TP nutzt und bestimmte Dienste innerhalb einer Anwendung erbringt. In openUTM ist das die Folge von Teilprogrammen, die einen Vorgang bilden. Ein TPSU-title ist ein eindeutiger Name innerhalb einer Anwendung. In openUTM ist dies der TAC-Name des ersten Teilprogramms eines Vorgangs. Der *initiating TPSU-title* ist der TPSU-title des Auftraggebers und der *recipient TPSU-title* der TPSU-title des Auftragnehmers.

openUTM unterstützt den in einer ISO-Norm definierten *Application Entity Title* (AET). Er wird dann benötigt, wenn mit Transaktionssicherung (Commit Functional Unit) gearbeitet wird, oder aber ein heterogener Partner einen AET für den Verbindungsaufbau erwartet. In openUTM wird der AET zusätzlich angegeben, für die Adressierung des Partners hat er keine Bedeutung.

Für jeden fernen Partner, mit dem die UTM-Anwendung über das OSI TP-Protokoll kommunizieren will, muss der *Application Context* definiert werden, der bei der Kommunikation mit diesem Partner verwendet werden soll.

Die OSI-Begriffe *Application Entity Title* und *Application Context* werden in den beiden folgenden Abschnitten etwas ausführlicher beschrieben.

Application Entity Title (AET)

In der OSI-Welt werden die Kommunikationspartner durch Anwendungsinstanzen (Application Entities) repräsentiert. Eine Anwendungsinstanz ist eine adressierbare Einheit in Schicht 7 des OSI-Referenzmodells (Anwendungsschicht). Eine solche Anwendungsinstanz ist z.B. der Zugriffspunkt (Access Point) einer UTM-Anwendung, über den sich ein OSI TP-Kommunikationspartner an die UTM-Anwendung binden kann. In der OSI TP-Norm wird jeder Anwendungsinstanz ein Application Entity Title zugeordnet, über den die Anwendungsinstanz im OSI-Netz eindeutig adressierbar ist.

In der ISO-Norm sind zwei Formen des AET definiert, die Directory-Form und die Object-Identifier-Form. openUTM unterstützt die Object-Identifier-Form des AET. Er wird dann benötigt, wenn mit Transaktionssicherung (Commit Functional Unit) gearbeitet wird, oder wenn ein heterogener Partner einen AET für den Verbindungsaufbau erwartet. Bei homogener UTM-UTM-Kommunikation wird der AET zusätzlich angegeben, für die Adressierung des Partners hat er keine Bedeutung. Ein AET besteht aus zwei Teilen:

– Application Process Title (APT)
– Application Entity Qualifier (AEQ)

Application Process Titel (APT)

Der APT wird zur Kennzeichnung der Anwendung verwendet. Der APT sollte gemäß der ISO-Norm global, d.h. weltweit eindeutig sein. Aus diesem Grund sollte er von einem Standardisierungsgremium vergeben und registriert werden, z.B. in Deutschland von der deutschen Gesellschaft für Warenkennzeichnung GmbH (DGWK).

Ein APT in Object-Identifier-Form besteht aus maximal 10 Komponenten:

```
(komponente1,komponente2,...,komponente10)
```

Die Werte für *komponente1* bis *komponente10* sind bereits teilweise standardisiert. Hierbei wurde einigen Zahlen ein symbolischer Name zugeordnet. Der Wertebereich für *komponente2* hängt vom Wert für *komponente1* ab. In der folgenden Tabelle sind die von openUTM unterstützten symbolischen Namen und die Wertebereiche dargestellt:

komponente1	`0:CCITT`	`1:ISO`	`2:JOINT-ISO-CCITT`

komponente2	`0:RECOMMENDATION`
	`1:QUESTION`
	`2:ADMINISTRATION`
	`3:NETWORK-OPERATOR`
		`0:STANDARD`
		`1:REGISTRATION-AUTHORITY`
		`2:MEMBER-BODY`
		`3:IDENTIFIED-ORGANIZATION`	

	erlaubte Werte: 0 - 39	erlaubte Werte: 0 - 39	erlaubte Werte: 0 - 67 108 863
komponente3 bis komponente10	erlaubte Werte: 0 - 67 108 863	erlaubte Werte: 0 - 67 108 863	erlaubte Werte: 0 - 67 108 863

Der APT, den Sie bei openUTM angeben, muss nicht von einem Standardisierungsgremium vergeben werden, d.h. Sie können den APT selbst vergeben. Er muss die beiden folgenden Anforderungen erfüllen:

– er muss innerhalb des Netzes eindeutig sein
– aus Werten bestehen, die gemäß der obigen Tabelle zulässig sind

Application Entity Qualifier (AEQ)

Der AEQ identifiziert einen Zugriffspunkt innerhalb einer Anwendung. Den Zugriffspunkten einer Anwendung können Sie nur dann AEQs zuordnen, wenn Sie der Anwendung selbst einen APT zugeordnet haben.

Der AEQ ist eine positive ganze Zahl zwischen 0 und 67 108 863.

Denselben AEQ dürfen Sie innerhalb einer Anwendung nicht mehrfach verwenden, d.h. in einer Anwendung dürfen nie zwei Zugriffspunkte mit demselben AEQ existieren. Sie müssen jedoch nicht allen Zugriffspunkten in einer Anwendung einen AEQ zuordnen.

Bei parallelen Associations wird beim Verbindungsaufbau geprüft, ob der AEQ derselbe ist wie bei der ersten aufgebauten Association.

Application Context

Mit jeder Partner-Anwendung, mit der Ihre lokale Anwendung über das OSI TP-Protokoll kommuniziert, muss der Application Context abgestimmt werden, der bei der Kommunikation verwendet werden soll.

Der Application Context muss für jede Partner-Anwendung explizit festgelegt werden. Der Application Context legt die Regeln fest, nach denen die Nachrichten zwischen der lokalen Anwendung und der Partner-Anwendung übertragen werden. openUTM unterstützt die folgenden vordefinierten Application Contexts:

– UDTAC
– UDTDISAC
– XATMIAC
– UDTCCR
– UDTSEC
– XATMICCR

Wenn Sie nicht einen der oben aufgeführten Standard-Application Contexts verwenden, können Sie weitere Application Contexts mit der Anweisung APPLICATION CONTEXT auf Seite 283 generieren.

Für einen Application Context vereinbaren alle beteiligten Partner Folgendes:

- Eine *Abstrakte Syntax*, die festlegt, wie die Benutzerdaten für die Übertragung codiert werden. openUTM unterstützt standardmäßig die folgenden Abstrakten Syntaxen:

  – UDT (Unstructured Data Transfer)
  – XATMI
  – CCR
  – UTMSEC

  Sehen Sie dazu auch die Anweisung ABSTRACT SYNTAX auf Seite 273.

- Eine *Transfersyntax* (Transfer Syntax), die festlegt, in welcher Form die Benutzerdaten übertragen werden. openUTM unterstützt standardmäßig die Transfersyntax Basic Encoding Rules (BER).

  Sehen Sie dazu auch die Anweisung TRANSFER SYNTAX auf Seite 536.

Beide Kommunikationspartner müssen dieselben Abstrakten Syntaxen als Application Context für die Kommunikation generieren. Wenn der generierte Application Context nicht mit dem im Partner generierten Application Context übereinstimmt, lehnt openUTM den Aufbau der Association mit einer entsprechenden Meldung ab.

Die Anweisungen ABSTRACT-SYNTAX, TRANSFER-SYNTAX und APPLICATION-CONTEXT werden nur benötigt, wenn Sie keinen der Standard-Application Contexts verwenden, die openUTM zur Verfügung stellt.

4.2.2 Generierung für die verteilte Verarbeitung über OSI TP

Für die Generierung der Kommunikationspartner und der Verbindungen zu diesen Partnern verfügen Sie über die folgenden KDCDEF-Anweisungen:

Anweisung	Funktion
ABSTRACT-SYNTAX	Abstrakte Syntax für die Benutzerdaten definieren: – Vergabe eines eindeutigen Object Identifiers – Transfersyntax für die Datenübertragung zuordnen
ACCESS-POINT	Adresse und Name des lokalen OSI TP-Zugriffspunktes definieren: – Application-Entity-Qualifier (AEQ) der lokalen Anwendung definieren (Adresskomponente des Application Entity Title)
APPLICATION-CONTEXT	Anwendungskontext für die Kommunikation mit der Partner-Anwendung definieren: – Abstrakte Syntax für die Benutzerdaten zuordnen – Vergabe eines eindeutigen Object Identifiers
LTAC	Lokale TAC-Namen für Services in der Partner-Anwendung vergeben, über die die fernen Services in der lokalen Anwendung angesprochen werden
MASTER-OSI-LPAP	Name und Eigenschaften des Master-LPAP in einem OSI-LPAP-Bündel festlegen (siehe Seite 100)
OSI-CON	Verbindungen zwischen der lokalen Anwendung und dem fernen Partner definieren und dem OSI-LPAP-Partner zuordnen: – Lokalen OSI TP-Zugriffspunkt angeben – Netzadresse der Partner-Anwendung angeben
OSI-LPAP	OSI-LPAP-Partner als logischen Anschlusspunkt für die Partner-Anwendung definieren: – Application Entity Title, d.h. APT und AEQ der Partner-Anwendung angeben – Application Context der Partner-Anwendung angeben – Anzahl der (parallelen) Verbindungen zum Partner und Namen der Verbindungen festlegen – Anzahl der Verbindungen, die beim Anwendungsstart automatisch aufgebaut werden – Anzahl der Verbindungen, für die die lokale Anwendung Contention Winner sein soll – Zugriffsrechte der Partner-Anwendung in der lokalen Anwendung – Administrationsberechtigungsstufe der Partner-Anwendung – Maximalwerte für Message Queue des OSI-LPAP-Partners festlegen – Status des OSI-LPAP-Partners bei Verbindungsaufbau – ggf. das OSI-LPAP zum Slave-LPAP eines OSI-LPAP-Bündels machen und das zugehörige Master-LPAP angeben

	Anweisung	Funktion
	TRANSFER-SYNTAX	Transfersyntax für die Datenübertragung definieren: – Vergabe eines eindeutigen Object Identifiers
	UTMD	Anwendungsglobale Werte und Adresse der lokalen UTM-Anwendung festlegen – Application-Process-Title (APT) definieren (Adresskomponente des Application Entity Title) – Maximale Wartezeit für Associationaufbau – Wartezeit für Quittung bei Asynchron-Nachrichten
X/W	**Unter Unix- und Windows-Systemen stehen zusätzlich folgende Parameter zur Verfügung:**	
X/W	MAX XAPTPSHMKEY	Schlüssel für das XAPTP Shared Memory Segment.
X/W	MAX OSISHMKEY	Schlüssel für das Shared Memory Segment von OSS definieren
X/W	MAX OSI-SCRATCH-AREA	Arbeitsbereichgröße für die dynamische Ablage von Daten festlegen

Für die Kommunikation mit dem OSI TP-Protokoll müssen Sie:

- Application Entity Title (AET) generieren.

  Mit der Anweisung UTMD...,APPLICATION-PROCESS-TITLE= legen Sie den Application Process Title (APT) als Adresskomponente des AET für Ihre Anwendung fest. Ein ferner Partner, der AETs verlangt, muss diesen APT kennen, um eine Verbindung aufbauen zu können.

  Mit der Anweisung ACCESS-POINT...,APPLICATION-ENTITY-QUALIFIER= legen Sie den AEQ als Adresskomponente des AET für den Zugriffspunkt der lokalen Anwendung fest. Eine Partner-Anwendung muss den AEQ des Zugriffspunkts kennen, über den die Kommunikation mit der lokalen Anwendung erfolgen soll.

  Den AET ordnen Sie dem OSI-LPAP-Partner zu. In der Anweisung OSI-LPAP geben Sie mit den Operanden APPLICATION-PROCESS-TITLE= den APT und mit APPLICATION-ENTITY-QUALIFIER= den Application Entity Qualifier (AEQ) des Zugriffspunkts für die Partner-Anwendung an. Der AEQ muss in der fernen Partner-Anwendung für den Zugriffspunkt generiert worden sein.

- Application Context für den Datenaustausch mit der Partner-Anwendung definieren.

  Den Application Context, der bei der Kommunikation mit der Partner-Anwendung verwendet werden soll, generieren Sie mit der Anweisung APPLICATION-CONTEXT, falls Sie nicht mit einem der UTM-Standard-Application Contexts auf Seite 93 arbeiten. Sie ordnen dabei dem Application Context definierte Abstrakte Syntaxen und einen eindeutigen Object Identifier zu.

  Die Abstrakte Syntax für die Übertragung der Benutzerdaten wird mit der Anweisung ABSTRACT-SYNTAX definiert, dabei wird ihr ein eindeutiger Object Identifier zugewiesen.

Die Transfersyntax, die festlegt, wie die Benutzerdaten für die Datenübertragung encodiert bzw. decodiert werden, definieren Sie mit der Anweisung ABSTRACT-SYNTAX. Identifiziert wird die Transfersyntax durch einen eindeutigen Object Identifier.

- Zugriffspunkt auf die Dienste von OSI TP für Ihre UTM-Anwendung definieren, damit Ihre Anwendung bei der Kommunikation über OSI TP adressierbar ist.

  Mit der Anweisung ACCESS-POINT wird die Adresse des Zugriffspunktes innerhalb des lokalen Systems angegeben. Mit ACCESS-POINT wird dem Zugriffspunkt auch ein symbolischer Name zugeordnet, über den der Zugriffspunkt in der lokalen UTM-Anwendung angesprochen werden kann.

  Die in ACCESS-POINT festgelegte Adresse muss innerhalb der UTM-Anwendung und innerhalb des lokalen Systems (unter BS2000/OSD) pro Host eindeutig sein. Sie ist deshalb mit dem Netzverwalter bzw. Systemverwalter Ihres Systems abzusprechen.

  Eine Partner-Anwendung, die mit der lokalen Anwendung über das OSI TP-Protokoll kommunizieren will, identifiziert die lokale UTM-Anwendung über die Adresse des Zugriffspunktes und die Netzadresse Ihres Systems. Die Netzadresse des Zugriffspunktes muss bei der Generierung der fernen Partner-Anwendungen angegeben werden.

- Logischen Anschlusspunkt (OSI-LPAP-Partner) für die Partner-Anwendung und die Verbindungen zwischen der lokalen Anwendung und ihr beschreiben.

  Die Beschreibung erfolgt über eine OSI-LPAP- und eine OSI-CON-Anweisung. Pro Partner-Anwendung muss eine OSI-LPAP-Anweisung generiert werden. Die Verbindungen zwischen der UTM-Anwendung und dem Kommunikationspartner werden in der OSI-CON-Anweisung definiert. Die Definition erfolgt über die beiden Zugriffspunkte (in der lokalen und in der Partner-Anwendung), zwischen denen die Verbindungen aufgebaut werden sollen. Dazu geben Sie in der OSI-CON-Anweisung die Netzadresse des fernen Zugriffspunktes und den Namen des lokalen Zugriffspunktes (in ACCESS-POINT definiert) an. Die Anzahl und die Namen der parallelen Verbindungen (Associations) zu der Partner-Anwendung legen Sie mit der Anweisung OSI-LPAP fest. Die Adresse des fernen Zugriffspunktes muss mit der Adresse des in der Partner-Anwendung generierten Zugriffspunktes übereinstimmen.

  Sollen Verbindungen automatisch aufgebaut werden, sobald beide Kommunikationspartner verfügbar, also gestartet sind, so muss bei der Generierung **beider** Partner angegeben werden, dass die Verbindungen automatisch beim Start der Anwendung aufgebaut werden sollen. Bei openUTM generieren Sie dies mit der Anweisung OSI-LPAP...,CONNECT=. Der Partner der zuletzt gestartet wird, baut dann die Verbindung auf. Mit OSI-LPAP...,CONTWIN= legen Sie auch fest, für wieviele Verbindungen zu dem Kommunikationspartner die lokale Anwendung Contention Winner sein soll.

- Den Services der Partner-Anwendungen lokal Transaktionscodes zuordnen, über die die fernen Services in der lokalen Anwendung angesprochen werden.

  Jeder dieser Transaktionscodes wird mit einer Steueranweisung LTAC definiert. Der Transaktionscode kann mit LTAC..., LPAP=*osi-lpap-name* der Partner-Anwendung eindeutig zugeordnet werden. Zusätzlich muss mit LTAC...,RTAC= der Transaktionscode des Teilprogramms in der Partner-Anwendung angegeben werden. Diesen Transaktionscode müssen Sie vom Betreiber der Partner-Anwendung erfragen. Falsche Angaben für Namen und Art des fernen TACs erkennt das Generierungstool KDCDEF nicht, da es keine Informationen über die Konfiguration der Partner-Anwendung hat. Ein Fehler tritt erst dann auf, wenn die lokale Anwendung diesen LTAC anfordert.

Zusätzlich können Sie für die Kommunikation über OSI TP:

- Globale Werte für alle Verbindungen der Anwendung zu Kommunikationspartnern festlegen

  Mit der Anweisung UTMD können Sie Wartezeiten auf Quittungen vom Kommunikationspartner begrenzen und festlegen, wieviele Aufträge an Partner-Anwendungen insgesamt (über OSI TP und LU6.1) gleichzeitig in der lokalen Anwendung bearbeitet werden dürfen. Mit entsprechenden Grenzwerten können Sie vermeiden, dass Verbindungen blockiert, bzw. Verbindungen zu früh abgebrochen werden, und dass alle Tasks der Anwendung durch Aufträge ferner Anwendungen belegt sind. Die festgelegten Werte gelten auch für die Kommunikation über LU6.1.

- Ersatzverbindungen generieren

  Ersatzverbindungen generieren Sie, indem Sie eine Verbindung mit zwei OSI-CON-Anweisungen beschreiben. Ersatzverbindungen müssen Sie generieren, wenn Sie mit verschiedenen Partnern alternativ zusammenarbeiten möchten, ohne in den Teilprogrammen darauf Rücksicht nehmen zu müssen. Die beiden Partner können sich in unterschiedlichen Systemen befinden. Ersatzverbindungen generieren Sie, indem Sie einer OSI-LPAP-Anweisung zwei OSI-CON-Anweisungen mit unterschiedlichen Partneradressen (Adressen des fernen Zugriffspunktes) zuordnen. So ordnen Sie zwei verschiedenen Partnern den gleichen logischen Anschlusspunkt (LPAP-Partner) zu. Sie dürfen jedoch nicht beide Verbindungen zu den alternativen Partner-Anwendungen gleichzeitig aktiv setzen. Deshalb darf nur jeweils in einer OSI-CON-Anweisung ACTIVE=YES angegeben werden. Umschalten auf die Ersatzverbindung zur alternativen Partner-Anwendung können Sie mit dem Administrationskommando KDCLPAP.

- Zugriffsschutz für Services der Partner-Anwendung definieren

  Mit der Anweisung LTAC..., LOCK= oder LTAC..., ACCESS-LIST= können Sie einen Service der Partner-Anwendung mit einem Zugriffsschutz versehen. Es können lokal dann nur Teilprogramme diesen Service adressieren, die unter einer Benutzerkennung (KCBENID) und von einem Client (KCLOGTER) aus gestartet wurden, die die entsprechenden Zugriffsrechte haben.

- Zugriffsschutz für Services der lokalen Anwendung definieren

  Soll die Partner-Anwendung nur bestimmte Services der lokalen Anwendung aufrufen können, dann schützen Sie kritische Services in der Anweisung TAC mit einem Lockcode oder einer Access-Liste. Mit einer KSET-Anweisung können Sie ein Keyset definieren, das die Keycodes für Services enthält, die die Partner-Anwendung nutzen darf. Das Keyset ordnen Sie dem logischen Anschlusspunkt der Partner-Anwendung mit der Anweisung OSI-LPAP...,KSET= zu. Die Partner-Anwendung darf dann nur die TACs aufrufen, die entweder nicht gesichert sind oder zu denen die Partner-Anwendung die entsprechenden Zugriffsrechte besitzt.

- Administrationsberechtigung für TACs der Partner-Anwendung vergeben

  In der OSI-LPAP-Anweisung können Sie festlegen, ob der Partner in der lokalen Anwendung Administrationsberechtigung erhalten soll. Die Berechtigungsstufe wird mit OSI-LPAP...,PERMIT= festgelegt.

B - UTM-SAT-Administrationsberechtigung für TACs der Partner-Anwendung vergeben

B Mit OSI-LPAP...,PERMIT= können Sie auch festlegen, ob der Partner in der lokalen An-
B wendung UTM-SAT-Administrationsberechtigung erhalten soll.

Die Grafik auf Seite 99 fasst zusammen, in welchen Punkten die Generierung der lokalen Anwendung und die Generierung der Partner-Anwendung aufeinander abgestimmt werden müssen. Es wird ein von openUTM generierter Standard-Application-Context verwendet, deshalb muss keine APPLICATION-CONTEXT-Anweisung geschrieben werden.
Weitere Informationen zur Generierung von Anwendungen mit verteilter Verarbeitung insbesondere über OSI TP können Sie der Beispielgenerierung „*Comfo*TRAVEL“ ab Seite 559 entnehmen.

Lokale Anwendung
Reservation Management Service

```
            .
            .
ACCESS-POINT ACRMS,
  P-SEL=(C'PRMS',ASCII),                    (TS)
  S-SEL=(C'SRMS',ASCII),                    (TS)
  T-SEL=C'RMS' ,                            (TS)
  APPLICATION-ENTITY-QUALIFIER=11           (1)



OSI-CON CNAGENCY,
  P-SEL=*NONE,
  S-SEL=*NONE,
  T-SEL=C'TRAVEL',                          (TS)
  N-SEL=C'HOST0002' ,                       (TS)
  LOCAL-ACCESS-POINT=ACRMS,
  OSI-LPAP=LPAGENCY

OSI-LPAP LPAGENCY,
  APPLICATION-CONTEXT=UDTCCR,
  APPLICATION-ENTITY-QUALIFIER=21,          (2)
  APPLICATION-PROCESS-TITLE =(1,2,3,20),    (3)
  ASSOCIATIONS=4,
  ASSOCIATION-NAMES=AGENCY,
  CONNECT=2,
  CONTWIN=1

LTAC LTAGENCY,
  LPAP  = LPAGENCY,
  RTAC = TCAGENCY,                          (4)
  WAITTIME=(10,30)

UTMD MAXJR=200,
  APPLICATION-PROCESS-TITLE =(1,2,3,10),    (5)
  CONCTIME=25,
  PTCTIME=0
            .
            .
            .
```

Partner-Anwendung
Travel Agency

```
            .
            .
OSI-CON CNRMS,
  P-SEL=(C'PRMS',ASCII) ,                   (TS)
  S-SEL=(C'SRMS',ASCII) ,                   (TS)
  T-SEL=C'RMS',                             (TS)
  N-SEL=C'HOST0001' ,                       (TS)
  LOCAL-ACCESS-POINT=ACTRAVEL,
  OSI-LPAP=LPRMS

ACCESS-POINT ACTRAVEL,
  P-SEL=*NONE,
  S-SEL=*NONE,
  T-SEL=C'TRAVEL',                          (TS)
  APPLICATION-ENTITY-QUALIFIER=21           (2)



OSI-LPAP LPRMS,
  APPLICATION-CONTEXT=UDTCCR,
  APPLICATION-ENTITY-QUALIFIER=11,          (1)
  APPLICATION-PROCESS-TITLE=(1,2,3,10),     (5)
  ASSOCIATIONS=4,
  ASSOCIATION-NAMES=RMS,
  CONNECT=2,
  CONTWIN=3

TAC TCAGENCY,                               (4)
  PROGRAM=PRAGENCY



UTMD MAXJR=200,
  APPLICATION-PROCESS-TITLE=(1,2,3,20),     (3)
  CONCTIME=25,
  PTCTIME=0
            .
            .
            .
```

(*n*) gibt an, welche Werte zwischen den beiden OSI TP-Generierungen abgestimmt werden müssen.

Bild 9: Abstimmung bei der Generierung von OSI TP-Anwendungen

4.2.3 OSI-LPAP-Bündel

OSI-LPAP-Bündel ermöglichen eine automatische Verteilung von Nachrichten auf mehrere OSI-LPAP-Partner. Soll eine UTM-Anwendung sehr viele Nachrichten mit einer Partner-Anwendung austauschen, kann es für die Lastverteilung sinnvoll sein, mehrere Instanzen der Partner-Anwendung zu starten und die Nachrichten auf die einzelnen Instanzen zu verteilen. In einem OSI-LPAP-Bündel übernimmt openUTM die Verteilung der Nachrichten an die Instanzen der Partner-Anwendung. Dazu müssen die Teilprogramme im APRO-Aufruf das MASTER-OSI-LPAP adressieren.

Ein Anwendungsfall für eine solche Verteilung der Nachrichten ist z.B. die Kommunikation einer UTM-Anwendung über BeanConnect mit einem J2EE Application Server. Wird der Application Server als Cluster-Anwendung betrieben, sollten die Nachrichten an den Application Server auf die einzelnen Instanzen des Clusters verteilt werden (siehe auch Handbuch „BeanConnect for openUTM“).

Ein weiterer Anwendungsfall ist die Kommunikation einer stand-alone UTM-Anwendung mit einer UTM-Cluster-Anwendung. Die Nachrichten an die UTM-Cluster-Anwendung können somit auf die einzelnen Knoten-Anwendungen verteilt werden.

Die Kopplung einer stand-alone UTM-Anwendung mit einer UTM-Cluster-Anwendung über ein OSI-LPAP-Bündel funktioniert analog zur Kommunikation zwischen zwei stand-alone UTM-Anwendungen mit OSI-LPAP-Bündel. Es gibt dabei keine Cluster-spezifischen Besonderheiten zu beachten.

Ein OSI-LPAP-Bündel besteht aus einem Master-LPAP und mehreren Slave-LPAPs. Die Slave-LPAPs werden dem Master-LPAP bei der Generierung zugeordnet. Die OSI-CONs von verschiedenen Slave-LPAPs adressieren dabei die verschiedenen Partner-Anwendungen.

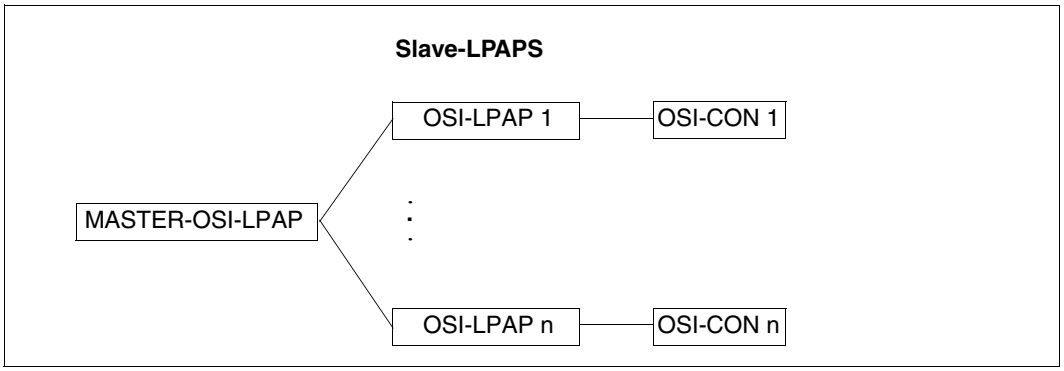


Bild 10: Beispiel OSI-LPAP-Bündel

OSI-LPAP-Bündel generieren

MASTER-OSI-LPAP-Anweisung auf Seite 367
Legt den Namen und die Eigenschaften des Master-LPAP in einem OSI-LPAP-Bündel fest:

- *master-lpap-name*

  Name für das Master-LPAP.

- APPLICATION-CONTEXT=

  Application Context, der für die Kommunikation mit dem entfernten Partner genutzt werden soll.

- STATUS=

  gibt an, ob Nachrichten an dieses LPAP-Bündel gesendet werden können.

OSI-LPAP-Anweisung auf Seite 429
Für die Generierung eines Slave-LPAP müssen die folgenden Eigenschaften festgelegt werden:

- *lpap-name*

  Name des Slave-LPAP.

- BUNDLE=master-lpap-name

  Name des Master-LPAP. Das Master-LPAP, das hier angegeben wird, muss in einer Anweisung MASTER-OSI-LPAP definiert sein. Durch Angabe von BUNDLE wird dieses OSI-LPAP zum Slave-LPAP des angegebenen Master-LPAP.

  ```
  MASTER-OSI-LPAP master, ...
  OSI-LPAP slave-lpap, BUNDLE=master, ...
  ```

- APPLICATION-CONTEXT=

  Application Context, der für die Kommunikation mit dem entfernten Partner genutzt werden soll.

  Allen Slave-LPAPs eines LPAP-Bündels muss der gleiche Application Context wie dem Master-LPAP zugeordnet sein.

OSI-CONs von LPAPs eines OSI-LPAP-Bündels

- Einem Master-LPAP dürfen keine physikalischen Verbindungen (OSI-CONs) zugeordnet werden. Es darf also nicht als OSI-LPAP in einer OSI-CON-Anweisung angegeben werden. Das Master-LPAP nutzt immer die Verbindungen, die den Slave-LPAPs zugeordnet sind.
- Alle OSI-CONs aller Slave-LPAPs eines LPAP-Bündels müssen dem gleichen lokalen ACCESS-POINT zugeordnet sein.

Verteilung von Nachrichten

Die nachfolgenden Informationen zur Verteilung von Nachrichten gelten in gleicher Weise für LU6.1 und OSI TP.

Teilprogramme können mit dem APRO-Aufruf sowohl ein Slave-LPAP als auch ein Master-LPAP adressieren. APRO-Aufrufe an ein Slave-LPAP werden von openUTM nicht weiter verteilt. APRO-Aufrufe an ein Master-LPAP verteilt openUTM wie folgt:

- Über APRO-Aufrufe, die an ein Master-LPAP gerichtet wurden, adressiert openUTM reihum die Slave-LPAPs.
  - openUTM versucht dabei immer, ein Slave-LPAP zu finden, zu dem bereits eine Verbindung aufgebaut ist und, wenn eine Asynchron-Nachricht an den Partner gesendet werden soll (APRO AM), dessen Queue-Level noch nicht erreicht ist.
  - Ist der erste APRO-Aufruf an ein Master-LPAP in einer Transaktion ein APRO DM, liefert openUTM nur dann den Returncode 40Z/KD10, wenn zu keinem Slave-LPAP eine Verbindung besteht.
  - Ist der erste APRO-Aufruf an ein Master-LPAP in einer Transaktion ein APRO AM, wählt openUTM nur dann ein Slave-LPAP mit nicht aufgebauter Verbindung, wenn zu keinem der Slave-LPAPs bereits eine Verbindung besteht. In diesem Fall wird für das Slave-LPAP der Verbindungsaufbau angestoßen.
  - Bei der Suche nach einem Slave-LPAP mit aufgebauter Verbindung wird für jedes Slave-LPAP, zu dem keine Verbindung aufgebaut ist, ein Verbindungsaufbau initiiert.
- Alle APRO-Aufrufe, die innerhalb einer Transaktion an das MASTER-LPAP gerichtet werden, adressieren das gleiche Slave-LPAP.
  Deswegen kann ggf. ein APRO-Aufruf für eine zweite Nachricht an eine Partner-Anwendung abgelehnt werden, wenn z.B. in der Zwischenzeit der Queue-Level für das Slave-LPAP überschritten wurde oder die Verbindung verloren gegangen ist.

- Nachrichten, die bereits einem Slave-LPAP zugeordnet wurden, werden in der Folge nicht mehr an ein anderes Slave-LPAP umgehängt. Geht für Dialog-Nachrichten die Verbindung nach dem APRO-Aufruf verloren, wird die Dialog-Nachricht wie bei „normalen“ LPAPs verworfen und die Transaktion ggf. zurückgesetzt.

Anzeige im KB-Kopf

Die nachfolgenden Informationen zur Anzeige im KB-Kopf gelten in gleicher Weise für LU6.1 und OSI TP.

In Vorgängen, die für empfangene Nachrichten gestartet werden, zeigt openUTM im KB-Kopf immer den Namen des LTERM bzw. (OSI-)LPAP an, über das die Nachricht empfangen wurde.

Für LPAP-Bündel gilt also:
In Vorgängen, die für Nachrichten gestartet werden, die über einen Slave-LPAP empfangen wurden, wird im KB-Kopf der Name dieses Slave-LPAPs angezeigt und **nicht** der Name des Master-LPAPs.

Sie können sich mit INIT PU darüber informieren, ob das (OSI-)LPAP im KB-KOPF Slave-LPAP eines LPAP-Bündels ist und wie das Master-LPAP heißt.

4.3 UTM- und BCAM-Generierung abstimmen (BS2000/OSD)

Bei der verteilten Verarbeitung über LU6.1 und OSI TP werden für die Kommunikation zwischen den Anwendungen Netzverbindungen benötigt. Für diese Netzverbindungen benutzt openUTM die Dienste des Transportsystems BCAM.

Um diese Netzverbindungen aufbauen zu können, müssen beiden Kommunikationspartnern Adressen zugeordnet werden. Diese Adressen müssen im Netz eindeutig sein. Der für die Kommunikation gewünschte Netztyp wird über die Angaben in der KDCDEF-Generierung (Anweisung BCAMAPPL T-PROT=) sowie in der BCAM-Generierung festgelegt. Hierbei ist insbesondere zu beachten, dass in openUTM nicht zwischen ISO- und TCP/IP-Netzen unterschieden wird (die Angaben ISO und RFC1006 beim Parameter T-PROT sind synonym). Diese Unterscheidung muss in der BCAM-Generierung bei der Definition der Verbindungen zwischen den beteiligten Rechnern getroffen werden.

Für eine UTM-Anwendung sind in der BCAM-Generierung i.A. keine Angaben erforderlich.

Bei einer Kopplung über RFC1006 nimmt BCAM als Listener-Portnummer des Partners standardmäßig die Nummer 102. Soll eine andere Portnummer verwendet werden, z.B. weil das Transportsystem des Partners die Portnummer 102 nicht verwenden kann, dann muss für die ferne Anwendung (SUBFUNCT=GLOBAL) ein BCMAP-Eintrag erzeugt werden. Dieser BCMAP-Eintrag wird folgendermaßen erstellt (siehe auch „BCAM Benutzerhandbuch“):

```
/BCMAP FUNCT=DEFINE,SUBFUNCT=GLOBAL,NAME=name-der-partner-anwendung, -
/      ES=partner-processorname,PPORT#=listener-port-nr,              -
/      PTSEL-I=tselektor-der-partner-anwendung
```

4.4 Adressinformationen für das Transportsystem CMX bereitstellen (Unix- und Windows-Systeme)

X/W Bei Verteilter Verarbeitung über LU6.1 oder OSI TP und bei der Anbindung von Clients vom Typ PTYPE=APPLI und UPIC-R verwendet openUTM das Transportzugriffssystem CMX (Unix- und Windows-Systeme). Bei der Kommunikation über CMX erfolgt die Anbindung über TCP/IP-RFC1006. openUTM holt die IP-Adressen der Partnerrechner beim Start aus der entsprechenden Datenbasis, z.B. der Hosts-Datei oder dem Domain Name Server.

X/W **Portnummer 102 bei TCP/IP-Verbindungen**

X/W Bei Verbindungen über TCP/IP-RFC1006 werden die Anwendungen über Portnummern angesprochen. Unabhängig davon, wie Sie die Adressinformationen bereitstellen, müssen Sie Folgendes beachten:

X Die Portnummer 102 kann nicht verwendet werden.

W In Windows-Systemen sollte die Portnummer 102 generell nicht verwendet werden. Der Grund liegt darin, dass in den genannten Fällen Port 102 häufig als privilegierter Port verwendet wird oder nur von einem einzigen Prozess auf dem Rechner verwendet werden kann.

4.4.1 Adressinformationen mit KDCDEF bereitstellen

X/W Die Adressinformationen werden in den Operanden PRONAM, N-SEL, LISTENER-PORT X/W sowie T-PROT und TSEL-FORMAT hinterlegt. Dabei müssen Sie Folgendes beachten:

X/W
X/W
X/W
X/W
X/W
X/W
X/W
- Bei PRONAM bzw. N-SEL **müssen** Sie immer den ersten TCP/IP-Hostnamen angeben, der in der Datenbasis (Hosts-Datei, DNS, ...) generiert ist, oder einen zugeordneten Mapped Hostname (siehe Abschnitt „Mapped Hostnamen verwenden (Unix- und Windows-Systeme)“ auf Seite 118). openUTM ermittelt beim aktiven Verbindungsaufbau aus dem Namen die IP-Adresse oder beim passiven Verbindungsaufbau aus der IP-Adresse den Namen. Weitere in der Datenbasis (Hosts-Datei, DNS,...) angegebene Namen, sogenannte Aliasnamen, dürfen Sie hier nicht angeben.

X/W
X/W
– Bei LISTENER-PORT sollten Sie immer eine Portnummer eintragen. Die Portnummer muss in jedem Fall mit dem Kommunikationspartner abgestimmt sein.

X/W
– Bei T-PROT müssen Sie immer RFC1006 angeben.

X/W
X/W
X/W
X/W
X/W
X/W
- Es wird empfohlen, den Operanden TSEL-FORMAT immer zu versorgen.
  Wenn Sie dort keine Angabe machen, dann vergibt KDCDEF abhängig von der OPTION-Anweisung folgenden Wert:
  – Bei CHECK-RFC1006=YES einen Standardwert anhand des Zeichenvorrats des zugehörigen Anwendungs- oder Partnernamens.
  – Bei OPTION CECK-RFC1006=NO den ungültigen Wert ’U’ bzw. ’?’ (= undefined)

X/W **OSI TP-Verbindung**

X/W
X/W
In folgendem Beispiel wird lokal die Portnummer `10000` und in der fernen Anwendung die Portnummer `12000` verwendet. Die ferne Anwendung läuft im Rechner `ZENTRAL1`.

```
X/W ACCESS-POINT BSPOSITP -
X/W ,LISTENER-PORT=10000 -
X/W ,T-PROT=RFC1006 -
X/W ,TSEL-FORMAT=T -
X/W ,P-SEL=.., ,S-SEL=... ,T-SEL=... -
X/W ,...
X/W
X/W OSI-CON OSICON01 -
X/W ,LOCAL-ACCESS-POINT=BSPOSITP -
X/W ,LISTENER-PORT=12000 -
X/W ,T-PROT=RFC1006 -
X/W ,N-SEL=ZENTRAL1 -
X/W ,P-SEL=... ,S-SEL=... ,T-SEL=... -
X/W ,...
```

X/W
X/W
Die Angaben bei P-SEL, S-SEL und T-SEL müssen mit der Generierung der Partner-Anwendung abgestimmt werden, siehe Beispiel auf Seite 99.

X/W **LU6.1-Verbindung**

X/W In folgendem Beispiel wird lokal die Portnummer `10010` und in der fernen Anwendung die
X/W Portnummer `12010` verwendet. Die ferne Anwendung läuft im Rechner `ZENTRAL2`.

```
X/W  BCAMAPPL BSPLU61 -
X/W  ,LISTENER-PORT=10010 -
X/W  ,T-PROT=RFC1006 -
X/W  ,T-SEL-FORMAT=T -
X/W  ,...
X/W
X/W  CON LU61PART -
X/W  ,BCAMAPPL=BSPLU61 -
X/W  ,PRONAM=ZENTRAL2 -
X/W  ,LISTENER-PORT=12010 -
X/W  ,T-PROT=RFC1006 -
X/W  ,TSEL-FORMAT=T -
X/W  ,...
```

X/W **PTYPE=APPLI-Verbindung**

X/W In folgendem Beispiel wird lokal die Portnummer `10020` und in der fernen Anwendung die
X/W Portnummer `12020` verwendet. Die ferne Anwendung läuft im Rechner `ZENTRAL3`.

```
X/W  BCAMAPPL BSPAPPLI -
X/W  ,LISTENER-PORT=10020 -
X/W  ,T-PROT=RFC1006 -
X/W  ,T-SEL-FORMAT=T -
X/W  ,...
X/W
X/W  PTERM APPLPART -
X/W  ,PTYPE=APPLI,
X/W  ,BCAMAPPL=BSPAPPL -
X/W  ,PRONAM=ZENTRAL3 -
X/W  ,LISTENER-PORT=12020 -
X/W  ,T-PROT=RFC1006 -
X/W  ,TSEL-FORMAT=T -
X/W  ,...
```

X/W **PTYPE=UPIC-R-Verbindung**

X/W In folgendem Beispiel wird lokal die Portnummer `10030` und in der fernen Anwendung die
X/W Portnummer `12030` verwendet. Die ferne Anwendung läuft im Rechner `ZENTRAL4`.

```
BCAMAPPL BSPUPR -
,LISTENER-PORT=10030 -
,T-PROT=RFC1006 -
,T-SEL-FORMAT=T -
,...

PTERM UPRPART -
,PTYPE=UPIC-R,
,BCAMAPPL=BSPUPR -
,PRONAM=ZENTRAL4 -
,LISTENER-PORT=12030 -
,T-PROT=RFC1006 -
,TSEL-FORMAT=T -
,...
```

X/W Beim KDCDEF-Aufruf werden standardmäßig Konsistenzprüfungen bzgl. der Adressie-
X/W rungs-Informationen durchgeführt. Dies erfolgt aufgrund der OPTION-Anweisung mit dem
X/W Operanden CHECK-RFC1006=YES.

4.4.2 Adressinformationen mit TNS-Einträgen bereitstellen

X/W Neben der UTM-Generierung kann die Bereitstellung der Adressinformation für das Transportsystem auch durch TNS-Einträge erfolgen. Welche der Adressinformationen während des Anwendungslaufs verwendet wird, steuern Sie mit der OPTION-Anweisung beim KDCDEF-Lauf. Sind beim KDCDEF-Lauf in der OPTION-Anweisung die Operanden CHECKTNS=NO und CHECK-RFC1006=YES angegeben (Standardeinstellung), wird nur die Adressinformation aus der UTM-Generierung verwendet.

X/W Andernfalls versucht openUTM zuerst, die Adressinformation im TNS-Directory zu finden. Wenn ein passender TNS-Eintrag gefunden wird, wird dieser mit höchster Priorität verwendet. In diesem Fall wird die Adressinformation in der UTM-Generierung ignoriert.

X/W Wird jedoch kein TNS-Eintrag gefunden, wird die Adressinformation aus der UTM-Generierung verwendet.

X/W Es wird empfohlen die Adressinformation immer vollständig mit der UTM-Generierung zu hinterlegen.

X/W Damit die komplette Generierungsinformation einer UTM-Anwendung gemeinsam zentral verwaltet werden kann, sollten Sie bestehende TNS-Einträge durch Hinterlegung der Adressinformation bei der UTM-Generierung ersetzen. Sorgen Sie anschließend dafür, dass die TNS-Einträge aus dem TNS-Directory gelöscht werden.

X/W Bei Verwendung der Funktion „Mapped Hostnamen“ wird immer die Adressinformation aus der UTM-Generierung verwendet (siehe Abschnitt „Mapped Hostnamen verwenden (Unix- und Windows-Systeme)“ auf Seite 118).

4.4.3 Umstellung der Adressinformation von TNS-Einträgen auf KDCDEF

X/W Wenn Sie die Adressinformation bisher über TNS-Einträge bereitgestellt haben, haben Sie in der UTM-Generierung nur die Anwendungs- und ggf. Rechnernamen der Kommunikationspartner sowie die Namen der eigenen UTM-Anwendung eingetragen. Diese Namen mussten Sie im TNS als GLOBALE NAMEN angeben. Die Zuordnung Rechnername/IP-Adresse erfolgte durch den TNS-Eintrag.

X/W Bei der Umstellung von TNS -Einträgen auf KDCDEF müssen Sie zusätzlich nur die Portnummer angeben.

X/W Anhand der LU6.1 Verbindung aus dem Abschnitt „Adressinformationen mit KDCDEF bereitstellen“ (siehe Seite 107) wird im Folgenden beispielhaft dargestellt, welche Änderungen Sie bei der Umstellung vornehmen müssen:

X/W **LU6.1-Verbindung**

X/W In folgendem Beispiel verwendet die lokale Anwendung `BSPLU61` die Portnummer `10010` und die ferne Anwendung LU61PART die Portnummer `12010`. Die ferne Anwendung läuft im Rechner `ZENTRAL2`.

X/W *Vor der Umstellung*

X/W KDCDEF:

```
BCAMAPPL BSPLU61-
CON LU61PART -
,BCAMAPPL=BSPLU61 -
,PRONAM=ZENTRAL2
,...
```

X/W TNS-Einträge:

```
BSPLU61\
       TSEL RFC1006 T'BSPLU61'
       TSEL LANINET A'10010'

LU61PART.ZENTRAL2\
       TA RFC1006 adresse von ZENTRAL2 PORT 12010 T'LU61PART'
```

X/W *Nach der Umstellung*

X/W Bei der Umstellung müssen Sie

X/W
X/W
X/W
- für den lokalen Anwendungsnamen
  statt der Portnummer im TNS-Eintrag (TSEL LANINET) die Portnummer im Parameter LISTENER-PORT der KDCDEF-Anweisung BCAMAPPL angeben.

X/W
X/W
X/W
X/W
- für den entfernten Partner
  statt der Adresse und der Portnummer in der Transportaddresse des TNS-Eintrags (TA) den Prozessornamen des Partnerrechners im Parameter PRONAM und die Portnummer im Parameter LISTENER-PORT der KDCDEF-Anweisung CON angeben.

X/W
X/W Damit die IP-Adresse des Partnerrechners bestimmt werden kann, muss der Name `ZENTRAL2` im DNS bekannt sein.

X/W KDCDEF:

```
BCAMAPPL BSPLU61 –
,LISTENER–PORT=10010 –
,T–SEL–FORMAT=T –
,...

CON LU61PART –
,BCAMAPPL=BSPLU61 –
,PRONAM=ZENTRAL2 –
,LISTENER–PORT=12010 –
,TSEL–FORMAT=T –
,...
```

X/W **LU6.1-Verbindung unter Verwendung symbolischer Namen**

X/W In diesem Beispiel verwendet die lokale Anwendung `BSPLU61` die Portnummer `10010` und
X/W die fernen Anwendung `LU61PART` die Portnummer `12010`. Die ferne Anwendung läuft im
X/W Rechner `ZENTRAL2`.

X/W *Vor der Umstellung*

X/W KDCDEF:
```
BCAMAPPL LU61-
CON LU61 -
,BCAMAPPL=LU61 -
,PRONAM=PROLU61
,...
```

X/W TNS-Einträge:
```
LU61\
       TSEL RFC1006 T'BSPLU61'
       TSEL LANINET A'10010'

LU61.PROLU61\
       TA RFC1006 adresse von ZENTRAL2 PORT 12010 T'LU61PART'
```

X/W *Nach der Umstellung*

X/W Im Vergleich zum Beispiel „LU6.1-Verbindung“ auf Seite 110 müssen Sie zusätzlich den
X/W BCAMAPPL-Namen und den CON-Namen an die TSEL-Werte der TNS-Einträge anpas-
X/W sen. Sie müssen also statt der symbolischen Namen die realen Namen verwenden.

X/W Bei der Umstellung müssen Sie

- X/W den Namen der lokalen Anwendung
  X/W in der Anweisung BCAMAPPL in den Wert ändern, den Sie im lokalen TNS-Eintrag
  X/W (TSEL RFC1006) verwendet haben.
- X/W für den lokalen Anwendungsnamen
  X/W wie im Beispiel „LU6.1-Verbindung“ auf Seite 110 statt der Portnummer im lokalen
  X/W TNS-Eintrag (TSEL LANINET) die Portnummer im Parameter LISTENER-PORT der
  X/W KDCDEF-Anweisung BCAMAPPL angeben.
- X/W den Namen der entfernten Anwendung
  X/W in der Anweisung CON in den Wert ändern, den Sie im entfernten TNS-Eintrag (TA) ver-
  X/W wendet haben.
- X/W für den entfernten Partner
  X/W wie im Beispiel „LU6.1-Verbindung“ auf Seite 110 statt der Adresse und der Portnum-
  X/W mer in der Transportaddress des TNS-Eintrags (TA) den Prozessornamen des Partner-
  X/W rechners im Parameter PRONAM und die Portnummer im Parameter LISTENER-PORT
  X/W der KDCDEF-Anweisung CON angeben.

X/W Damit die IP-Adresse des Partnerrechners bestimmt werden kann, muss der Name
X/W ZENTRAL2 im DNS bekannt sein.

X/W KDCDEF:
```
BCAMAPPL BSPLU61 -
,LISTENER-PORT=10010 -
,T-SEL-FORMAT=T -
,...

CON LU61PART -
,BCAMAPPL=BSPLU61 -
,PRONAM=ZENTRAL2 -
,LISTENER-PORT=12010 -
,TSEL-FORMAT=T -
,...
```

4.5 Adressinformationen für das Transportsystem SOCKET bereitstellen (Unix- und Windows-Systeme)

X/W Bei Verbindungen zu einer TS-Anwendung mit PTYPE=SOCKET erfolgt die Kommunikation über die Socket-Schnittstelle mit native TCP/IP als Transportprotokoll. Für diese Socket-Clients kann die Adressinformation nur mit KDCDEF bereitgestellt werden. openUTM holt die IP-Adressen der Partnerrechner beim Start aus der entsprechenden Datenbasis, z.B. der Hosts-Datei oder dem Domain Name Server.

X/W Die Adressinformationen werden in den Operanden PRONAM und LISTENER-PORTsowie T-PROT und TSEL-FORMAT hinterlegt. Zusätzlich ist noch der Operand BCAMAPPL von Bedeutung.

X/W Beachten Sie dabei folgende Punkte:

- X/W Bei PRONAM müssen Sie immer den ersten TCP/IP-Hostnamen angeben, der in der Datenbasis (Hosts-Datei, DNS, ...) generiert ist, oder einen zugeordneten Mapped Hostname (siehe Abschnitt „Mapped Hostnamen verwenden (Unix- und Windows-Systeme)“ auf Seite 118). openUTM ermittelt daraus die IP-Adresse. Weitere in der Datenbasis (Hosts-Datei, DNS,...) angegebene Namen, sogenannte Aliasnamen, dürfen Sie hier nicht angeben.
- X/W Bei LISTENER-PORT müssen Sie immer eine Portnummer eintragen. Die Portnummer muss in jedem Fall mit dem Kommunikationspartner abgestimmt sein. Die Angabe von LISTENER-PORT ist Pflicht.
- X/W Bei T-PROT müssen Sie immer SOCKET angeben.
- X/W Es wird empfohlen, den Operanden TSEL-FORMAT immer zu versorgen.
- X/W Bei BCAMAPPL müssen Sie einen Anwendungsnamen angeben, für den T-PROT=SOCKET generiert wurde.

X/W **PTYPE=SOCKET-Verbindung**

X/W In folgendem Beispiel wird lokal die Portnummer `10100` und in der fernen Anwendung die
X/W Portnummer `12100` verwendet. Die ferne Anwendung läuft im Rechner `ZENTRAL5`.

```
X/W BCAMAPPL BSPSOC -
X/W ,LISTENER-PORT=10100 -
X/W ,T-PROT=SOCKET -
X/W ,T-SEL-FORMAT=T -
X/W ,...
```

```
X/W PTERM SOCPART -
X/W ,PTYPE=SOCKET,
X/W ,BCAMAPPL=BSPSOC -
X/W ,PRONAM=ZENTRAL5 -
X/W ,LISTENER-PORT=12100 -
X/W ,T-PROT=SOCKET -
X/W ,TSEL-FORMAT=T -
X/W ,...
```

4.6 Multi- und single-threaded Netzanbindung (Unix- und Windows-Systeme)

X/W Bei verteilter Verarbeitung werden UTM-Anwendungen über Netzprozesse ans Netz angebunden. Diese Prozesse haben die Aufgabe, Verbindungsanforderungen abzuwickeln und den Datenaustausch auf der Verbindung zu verwalten. Bei der Generierung können Sie festlegen, ob die UTM-Anwendung über multi-threaded Netzprozesse oder über single-threaded Netzprozesse verteilt arbeiten soll.

X/W Aus Performancegründen wird empfohlen die UTM-Anwendung mit multi-threaded Netzprozessen zu betreiben. Dies ist auch die Standardeinstellung.

X/W **Verwendung von multi-threaded und single-threaded Netzprozessen**

X Bei verteilter Verarbeitung über CMX-Verbindungen auf Unix-Systemen können Sie sowohl multi-threaded als auch single-threaded Netzprozesse verwenden.

W Bei verteilter Verarbeitung über CMX-Verbindungen auf Windows-Systemen können Sie nur multi-threaded Netzprozesse verwenden.

X/W Bei verteilter Verarbeitung über Socket-Verbindungen können Sie nur multi-threaded Netzprozesse verwenden.

X/W **Multi-threaded Netzprozess bei CMX- und Socket-Verbindungen**

X/W Die multi-threaded Netzanbindung erlaubt einen sparsamen Einsatz der Prozess-Ressourcen, da ein Netzprozess mehrere Verbindungen verwalten kann.

X/W Der Main-Prozess *utmmain* einer UTM-Anwendung startet einen oder mehrere Netzprozesse, in denen jeweils mehrere Verbindungen aufgebaut und verwaltet werden. Die Zuordnung zwischen Verbindungen und Prozessen wird über **Listener-IDs** geregelt. Alle Verbindungen mit derselben Listener-ID werden von demselben Netzprozess verwaltet.

X/W Für CMX- und für Socket-Verbindungen gibt es jeweils einen eigenen Typ von multi-threaded Netzprozess. Bei CMX-Verbindungen handelt es sich um den Prozesstyp *utmnet*, bei Socket-Verbindungen um den Prozesstyp *utmnets*.

X/W Multi-threaded Netzanbindung legen Sie in der **MAX-Anweisung** mit dem folgenden Operanden fest:

X/W
- NET-ACCESS=MULTI-THREADED

  X/W Mehrere gleichzeitig existierende Kommunikationsbeziehungen werden in einem Netzprozess bedient, Standardeinstellung.

X/W
X/W

Listener-IDs können Sie in der **ACCESS-POINT-Anweisung** für Zugriffspunkte und in der **BCAMAPPL-Anweisung** für Anwendungsnamen definieren.

X/W

- LISTENER-ID=number

X/W
X/W

Ordnet dem Zugriffspunkt oder Anwendungsnamen eine Listener-ID zu, als Verwaltungsinformation für die Prozesszuordnung bei multi-threaded Netzanbindung.

X/W
X/W
X/W

Aufgrund der unterschiedlichen Typen von multi-threaded Netzprozessen sind die Listener-Ids zu CMX Verbindungen und die Listener-IDs zu Socket-Verbindungen disjunkte Wertebereiche.

X/W
X/W
X/W
X/W
X/W

Die Zuordnung der Verbindung zu einem Netzprozess erfolgt über die dem lokalen Anwendungsnamen bzw. dem lokalen Zugriffspunkt zugeordnete Listener-ID. Wenn Sie für eine Anwendung die multi-threaded Netzanbindung eingestellt haben, dann ordnet openUTM allen Anwendungsnamen und Zugriffspunkten, für die Sie keine Listener-ID vergeben, die Listener-ID 0 zu. Alle diese Verbindungen werden dann von einem Netzprozess bedient.

X

Single-threaded Netzprozess (nur bei CMX-Verbindungen auf Unix-Systemen)

X
X
X
X
X

Der Mainprozess *utmmain* einer UTM-Anwendung startet einen Netz-Main-Prozess *utmnetm*. Dieser Prozess hat die Aufgabe, den Verbindungsaufbau vollständig für alle CMX-Verbindungen der UTM-Anwendung abzuwickeln. Dabei kann der Aufbauwunsch sowohl von der lokalen UTM-Anwendung als auch von einer fernen Partner-Anwendung aktiviert worden sein.

X
X
X
X
X
X

Kommt eine Verbindung zwischen den Partnern zustande, dann startet *utmnetm* speziell für diese Verbindung einen Netz-Kind-Prozess, den *utmnetc*-Prozess. Dieser Prozess verwaltet den Datenaustausch zwischen den Partnern und existiert bis zum Abbau der Verbindung. Pro bestehender Verbindung zwischen der UTM-Anwendung und deren Partner-Anwendungen existiert ein *utmnetc*-Prozess. Bei single-threaded Netzanbindung wird also jede Verbindung von einem eigenen Prozess verwaltet.

X
X

Single-threaded Netzanbindung legen Sie in der **MAX-Anweisung** mit dem folgenden Operanden fest:

X

- NET-ACCESS=SINGLE-THREADED

X

Jede Kommunikationsbeziehung wird in einem eigenen Netzprozess verwaltet.

X
X

Bei single-threaded Netzanbindung hat der Wert von Listener-ID, den Sie beim Zugriffspunkt oder Anwendungsnamen vergeben haben, keine Bedeutung.

4.7 Mapped Hostnamen verwenden (Unix- und Windows-Systeme)

X/W Für die Kommunikation von openUTM mit entfernten Partner-Anwendungen muss bei der Generierung eine eindeutige Adressinformation hinterlegt werden. Die Adressinformation enthält den UTM-Hostnamen des Partnerrechners und die Portnummer. Der UTM-Hostname ist in der Regel gleich dem IP-Hostnamen (realer Hostname). Optional kann ein Mapping zwischen einem UTM-Hostnamen und einem IP-Hostnamen erfolgen. Ein Mapping von Hostnamen ist dann sinnvoll, wenn

X/W
- reale Hostnamen mit mehr als 8 Zeichen vorliegen.

X/W
- der Hostname eines Partnerrechners für den aktiven Verbindungsaufbau ein anderer ist als der Hostname, der beim passiven Verbindungsaufbau geliefert wird (z.B. bei Verwendung einer Firewall).

X/W
- die unterschiedlichen Hostnamen in einem Cluster in der openUTM-Partner-Anwendung intern auf einen festen logischen Prozessornamen des Clusters abgebildet werden.

X/W openUTM führt nur dann ein Mapping von Hostnamen durch, wenn eine gültige Umwandlungsdatei existiert und die Umgebungsvariable UTM_NET_HOSTNAME gesetzt ist (siehe Abschnitte „Umwandlungsdatei für Mapped Hostnamen“ auf Seite 119 und „Umgebungsvariable UTM_NET_HOSTNAME“ auf Seite 121).

X/W **Generierung**

X/W Wenn Sie das Mapping von Hostnamen verwenden wollen, gehen Sie wie folgt vor:

X/W ► Erstellen Sie eine Umwandlungsdatei und legen Sie dort die Zuordnung von UTM-Hostnamen zu realen Hostnamen fest (siehe Seite 119).

X/W ► Setzen Sie die Umgebungsvariable UTM_NET_HOSTNAME, um die Umwandlungsdatei zu aktivieren (siehe Seite 121).

X/W ► Ändern Sie ihren Input für die Generierung so ab, dass Sie bei den jeweiligen KDCDEF Anweisungen als Werte für den Hostnamen nicht den realen Hostnamen sondern einen UTM-Hostnamen angeben.

X/W Falls die Umgebungsvariable UTM_NET_HOSTNAME mit einer gültigen Umwandlungsdatei gesetzt ist, ist die TNS-Funktionalität für die gesamte UTM-Anwendung deaktiviert.

X/W **Laufzeit**

X/W openUTM setzt einen UTM-Hostnamen zur Laufzeit in einen realen Hostnamen um und
X/W umgekehrt:

X/W
X/W
X/W
- Falls eine gültige Umwandlungsdatei vorliegt, wird bei einem aktiven Verbindungsaufbau versucht, den UTM-Hostname des Partnerrechners in einen realen Hostnamen umzuwandeln, bevor openUTM die zugehörige IP-Adresse ermittelt.

  X/W
  X/W
  Ist eine Umwandlung des Hostnamen nicht möglich, wird die IP-Adresse aus dem UTM-Hostnamen ermittelt.

X/W
X/W
X/W
- Bei einem passiven Verbindungsaufbau wird aus der IP-Adresse des Partners der reale Hostname ermittelt. Falls eine gültige Umwandlungsdatei vorliegt, wird versucht, den realen Hostnamen in einen UTM-Hostnamen umzuwandeln.

  X/W
  X/W
  – War diese Umwandlung erfolgreich, prüft openUTM die Generierungsinformation mit dem umgewandelten Hostnamen.

  X/W
  X/W
  – War diese Umwandlung nicht erfolgreich, prüft openUTM die Generierungsinformation mit dem realen Hostnamen.

X/W
X/W
X/W
X/W
X/W
X/W
X/W
X/W
i Ob für eine Verbindung eine Umwandlung des UTM-Hostnamens in einen realen Hostnamen bzw. umgekehrt vorgenommen wurde, erkennen Sie anhand des Wertes für den Prozessornamen im Insert `TNSNAME` in den Meldungen des Netzprozesses.
Falls eine Umsetzung stattgefunden hat, besteht der Wert für den Prozessornamen aus dem UTM-Hostnamen in Klammern, ergänzt um den realen Hostnamen z.B. `system1(zentrale)`. Falls keine Umsetzung stattgefunden hat, besteht der Wert für den Prozessornamen nur aus dem realen Hostnamen z.B. `zentrale`.

4.7.1 Umwandlungsdatei für Mapped Hostnamen

X/W Die Umwandlungsdatei enthält alle Vorschriften, nach denen UTM-Hostnamen in reale
X/W Hostnamen und umgekehrt umgewandelt werden.

X/W openUTM liest die Umwandlungsdatei beim Start einer UTM-Anwendung ein und generiert
X/W aus diesen Angaben eine interne Hostname-Tabelle. Die Informationen in dieser Tabelle
X/W werden zur Laufzeit für die Umwandlung der Hostnamen verwendet.

X/W openUTM stellt bei der Installation eine Musterdatei mit dem Namen *utmhostname* zur Ver-
X/W fügung. Sie finden die Datei direkt im *utmpfad* Verzeichnis von openUTM.

X/W
X/W
X/W
X/W
i In der Umwandlungsdatei muss nicht für alle generierten Hostnamen ein Eintrag existieren, sondern nur für Hostnamen, für die ein Mapping durchgeführt werden soll. Für alle generierten Hostnamen, für die keine Umwandlungsvorschrift existiert, wird die IP-Adresse aus dem UTM-Hostnamen bestimmt.

X/W **Dateiname**

X/W Sie können den Namen für die Umwandlungsdatei frei wählen.
X/W Standard: *utmhostname*

X/W Der Dateiname darf maximal 300 Zeichen lang sein.

X/W **Dateiformat**

X/W Die Umwandlungsdatei ist zeilenorientiert. Die Zeilen haben folgendes Format:

```
X/W # (#) openUTM Hostname File
X/W
X/W # Umwandlungsvorschriften
X/W
X/W utm_hostname_1␣real_hostname_1␣[real_hostname_2]␣...
X/W ....
X/W utm_hostname_n␣real_hostname_m␣[real_hostname_o]␣...
X/W ....
```

X/W `# (#) openUTM Hostname File`
X/W Header der Umwandlungsdatei. Der Header muss immer die erste Zeile in der Datei sein.

X/W *Leerzeile*
X/W Leerzeilen werden beim Einlesen ignoriert.

X/W *#Kommentar*
X/W Kommentarzeilen beginnen mit einem '#'-Zeichen in Spalte 1 und werden beim Einlesen ignoriert.

X/W *utm_hostname_x*
X/W UTM-Hostname, für den eine Umwandlung in einen realen Hostnamen und umgekehrt vorgenommen werden soll.

X/W Der UTM-Hostname muss in Spalte 1 beginnen.
X/W Maximale Länge: 8 Zeichen

X/W *real_hostname_*y
X/W Realer Hostname eines Partnerrechners, wie er im Name Service des lokalen Systems (z.B. der hosts-Datei) eingetragen ist. Sie können in einer Umwandlungsvorschrift mehrere reale Hostnamen, durch ein oder mehrere Leerzeichen/Tabulatoren getrennt, angeben.

X/W Wenn Sie hier einen realen Hostnamen angegeben haben, zu dem keine IP-Adresse gefunden werden kann, erhalten Sie zur Laufzeit der UTM-Anwendung eine K154-Meldung (IP-Adresse konnte nicht ermittelt werden).

X/W Maximale Länge: 64 Zeichen

X/W Für die Umwandlungsdatei gelten folgende Maximalwerte und Regeln:

X/W
- Maximale Anzahl der Umwandlungsvorschriften pro Datei: 500

X/W
- Eine Umwandlungsvorschrift endet am Zeilenende, es gibt keine Fortsetzungszeilen.

X/W
- Maximale Zeilenlänge: 300 Zeichen

X/W
- Maximal Anzahl reale Hostnamen in der Umwandlungsdatei: 2000

X/W
- Für einen UTM-Hostnamen können mehrere Umwandlungsvorschriften vorliegen.

X/W X/W
- Die Reihenfolge der Umwandlungsvorschriften in der Datei bestimmt zur Laufzeit die Priorität bei der Umwandlung der Hostnamen.

X/W
- Leerzeilen und Kommentarzeilen werden ignoriert.

X/W
- Syntaktisch fehlerhafte Zeilen werden ignoriert.

4.7.2 Umgebungsvariable UTM_NET_HOSTNAME

X/W X/W X/W X/W Die Umgebungsvariable UTM_NET_HOSTNAME gibt die Umwandlungsdatei für Hostnamen an, die der UTM-Anwendung zugewiesen wird (siehe openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter Unix- und Windows-Systemen“ unter „Allgemeine Umgebungsvariablen“).

X/W X/W Ist diese Variable gesetzt, wird die angegebene Umwandlungsdatei beim Starten einer UTM-Anwendung ausgewertet.

X/W X/W X/W X/W Die Umgebungsvariable enthält den vollständigen Dateinamen der Umwandlungsdatei inklusive der Pfadangabe. Wenn Sie UTM_NET_HOSTNAME nur setzen und keinen Dateinamen angeben, sucht openUTM die Datei *utmhostname* im lokalen Verzeichnis (Verzeichnis, in dem der utmmain-Prozess gestartet wurde).

5 Hinweise zur Generierung ausgewählter Objekte und Funktionen der Anwendung

In diesem Kapitel wird beschrieben, wie Sie einige Objekte Ihrer UTM-Anwendung konfigurieren und welche KDCDEF-Steueranweisungen bzw. welche der einzelnen Operanden für die Beschreibung der Objekte relevant sind. Dies gilt für die folgenden UTM-Objekte:

- Clients (Seite 124)
- B/X B/X B/X Drucker, Druckersteuer-LTERMs, Druckerbündel unter BS2000/OSD und Unix-Systemen (Seite 165) sowie RSO-Drucker an einer UTM-Anwendung unter BS2000/OSD (Seite 169)
- Service-gesteuerte Queues (Seite 178)
- Meldungsmodule (Seite 183)
- B Multiplexanschlüsse einer UTM-Anwendung unter BS2000/OSD (Seite 189)
- B B BLS-Lademodule und Common Memory Pools einer UTM-Anwendung unter BS2000/OSD (Seite 196)

Darüber hinaus ist hier für einige ausgewählte UTM-Funktionen beschrieben, welche KDCDEF-Steueranweisungen und welche ihrer Operanden für den Einsatz dieser Funktionen von Bedeutung sind. Dies gilt für folgende Funktionen:

- Auftragssteuerung über Prioritäten und Prozessbeschränkung (Seite 205)
- Zugriffsschutz-Funktionen (Seite 216)
- Verschlüsselung von Nachrichten auf Verbindungen zu Clients (Seite 225)
- Kopplung mit Resource Managern und Datenbanken (Seite 232)
- B Internationalisierung einer UTM-Anwendung unter BS2000/OSD (Seite 235)
- B Zugangskontrolle mit Kerberos unter BS2000/OSD (Seite 242)

Sehen Sie dazu auch das openUTM-Handbuch „Konzepte und Funktionen“.

5.1 Clients an die Anwendung anschließen

Dieser Abschnitt beschreibt die Generierung von Terminals, UPIC-Clients, Transportsystem-Anwendungen und OpenCPIC-Clients. Transportsystem-Anwendungen sind DCAM, PDN-, CMX- und Socket-Anwendungen sowie UTM-Anwendungen, die als Transportsystem-Anwendung generiert sind. Sie werden im Folgenden kurz TS-Anwendungen genannt.

Welche Möglichkeiten Ihnen UPIC-Clients bieten, finden Sie ausführlich im Handbuch „openUTM-Client für das Trägersystem UPIC“ beschrieben.

Jeder Client, der Services einer UTM-Anwendung nutzen will, muss der UTM-Anwendung bekannt sein. Ein Client ist einer UTM-Anwendung dann bekannt, wenn er einem in der Konfiguration definierten logischen Anschlusspunkt zugeordnet ist. Hier unterscheiden sich die einzelnen Typen von Clients:

- Bei Terminals, UPIC-Clients und TS-Anwendungen wird ein logischer Anschlusspunkt **LTERM-Partner** genannt. Für den Anschluss an LTERM-Partner gibt es zwei Möglichkeiten:
  - Sie generieren den Client für einen Einzelanschluss, indem Sie den physischen Client mit einer PTERM-Anweisung definieren und exklusiv einem LTERM-Partner zuordnen, siehe unten. Einen Client müssen Sie immer dann mit PTERM generieren, wenn von der UTM-Anwendung aus Verbindungen zu diesem Client aufgebaut werden sollen (z.B. zu TS-Anwendungen). Für andere Clients müssen Sie nur dann eine PTERM-Anweisung absetzen, wenn Sie ihnen bestimmte logische Eigenschaften zuordnen wollen, z.B. besondere Zugriffsrechte, die Sie keinem LTERM-Pool zuordnen wollen.
  - Sie definieren einen Pool von LTERM-Partnern, genannt LTERM-Pool, siehe Seite 129. Über einen LTERM-Pool können sich mehrere Clients anschließen.
- Bei OpenCPIC-Clients wird der logische Anschlusspunkt **OSI-LPAP-Partner** genannt. Über einen OSI-LPAP-Partner können mehrere parallele Verbindungen aufgebaut werden.

Die ersten beiden Abschnitte zeigen die grundlegenden Schritte, um einen Client anzuschließen. Die Abschnitte „Anmeldeverfahren für Clients definieren“ auf Seite 140 bis „Beispiele für die Generierung eines Client/Server-Verbundes“ auf Seite 160 gehen näher auf einzelne Aspekte wie z.B. Anmeldeverfahren, Security-Funktionen oder Adressierung ein.

5.1.1 Clients über LTERM-Partner anschließen

Wenn Sie physische Terminals, UPIC-Clients und TS-Anwendungen einzeln anschließen wollen, müssen Sie für jeden Client folgende Generierungsanweisungen geben:

- eine LTERM-Anweisung für den logischen Anschlusspunkt
- eine PTERM-Anweisung für den physischen Client

Zusätzlich ist bei UPIC-Clients und TS-Anwendungen eventuell eine BCAMAPPL-Anweisung notwendig. Grenzwerte, Maximalwerte und Parameter, die anwendungsweit für die Kommunikation zwischen Clients und UTM-Anwendung eingestellt werden sollen, werden in der MAX-Anweisung definiert.

LTERMs und PTERMs können auch dynamisch erzeugt werden (Objekt KC_LTERM und KC_PTERM). Zudem lässt sich die Zuordnung des Clients zum LTERM-Partner in der PTERM-Anweisung per Administration später dynamisch anpassen. Beispielsweise können Sie einem LTERM-Partner im laufenden Betrieb einen anderen Client (gleichen Typs) zuordnen, oder aber einem Client einen anderen LTERM-Partner zuordnen, für den Sie andere Zugriffsrechte definiert haben. Sehen Sie dazu das openUTM-Handbuch „Anwendungen administrieren“.

LTERM-Anweisung auf Seite 354
Die wichtigsten Eigenschaften für LTERM-Partner, über die sich Clients an eine Anwendung anschließen können, werden mit den folgenden Operanden festgelegt:

- *ltermname*

  Name des LTERM-Partners. Logischer Name, über den der Client, dem der LTERM-Partner zugeordnet ist, aus den Teilprogrammen der Anwendung heraus angesprochen wird.

- KSET=

  Keyset des LTERM-Partners, das heißt ein Berechtigungsprofil, das festlegt, welche Teile des Anwendungsprogramms (welche TACs) dem Client, der sich über diesen LTERM-Partner an die Anwendung anschließt, zur Verfügung stehen.

B

- LOCALE= (nur BS2000/OSD)

B
B
B
B

  LTERM-spezifische Sprachumgebung der Clients, die sich über diesen LTERM-Partner an die Anwendung anschließen. Diese Sprachumgebung wird von openUTM auch für die Ausgabe von Meldungen verwendet, solange noch kein Benutzer angemeldet ist.

- LOCK=

  Lockcode als Zugangsschutz. Die Verbindung wird nur hergestellt, wenn der Client sich mit einer Benutzerkennung bei openUTM anmeldet, für die ein Keyset generiert wurde, mit einem Keycode, der diesem Lockcode entspricht.

- USAGE=D

  Typ des Kommunikationspartners. In diesem Fall schließen sich Dialog-Partner über den LTERM-Partner an die Anwendung an. Nachrichtenaustausch ist in beide Richtungen möglich.

- USER=

  Benutzerkennung, unter der der Client nach dem Verbindungsaufbau automatisch angemeldet wird, siehe Abschnitt „Automatisches Anmelden unter einer bestimmten Benutzerkennung“ auf Seite 141. Für diese Benutzerkennung können Sie weitere Eigenschaften definieren, siehe Abschnitt „Securityfunktionen generieren“ auf Seite 145.

PTERM-Anweisung auf Seite 440
Die wichtigsten Eigenschaften für physische Clients werden mit den folgenden Operanden festgelegt:

- *ptermname*

  Name des Clients wie er im System der Server-Anwendung generiert ist.

B
B

  BS2000/OSD (ohne Socket-Anwendung):
  Es ist der BCAM-Name des Client anzugeben.

  Socket-Anwendungen:
  Wenn der Verbindungsaufbau nur von openUTM ausgehen soll, kann *ptermname* beliebig gewählt werden, andernfalls muss der Name die Form `PRT`*nnnnn* haben. Dabei ist *nnnnn* die Portnummer, über die die Socket-Anwendung die Verbindung aufgebaut hat, ggf. ergänzt um führende Nullen.

  Näheres siehe Abschnitt „Adressinformationen bereitstellen“ auf Seite 150.

- BCAMAPPL=

  Name der lokalen Anwendung, über den das Transportsystem die Verbindung zwischen Client und UTM-Anwendung aufbaut. Dieser Name muss in einer BCAMAPPL-Anweisung oder mit MAX ...APPLINAME= definiert sein. Wenn Sie diesen Operanden weglassen, wird der Name aus MAX ...APPLINAME= genommen. Terminals dürfen nur den Namen aus MAX ... APPLINAME= verwenden.

- ENCRYPTION-LEVEL=

  Für UPIC-Clients und unter BS2000/OSD zusätzlich für bestimmte Terminalemulationen legen Sie die minimale Verschlüsselungsebene fest, die auf Verbindungen zum Client eingehalten werden muss. Sie können den Client auch als vertrauenswürdig einstufen, d.h. dieser Client kann ohne Verschlüsselung mit der UTM-Anwendung arbeiten. Zur Verschlüsselung siehe auch Abschnitt „Nachrichten-Verschlüsselung auf Verbindungen zu Clients“ auf Seite 225.

- LTERM=

  Dem physikalischen Client wird als logischer Anschlusspunkt der LTERM-Partner *ltermname* zugeordnet, über den der Client sich an die UTM-Anwendung anschließt.

- PRONAM=

  Name des Rechners, auf dem sich der Client befindet.

- PTYPE=

  Typ des Clients, der sich über den LTERM-Partner anschließt. Sie geben hier an, ob der Client eine TS-Anwendung, ein UPIC-Client oder ein Terminal ist.

X/W

- T-PROT=, TSEL-FORMAT= (nur unter Unix- und Windows-Systemen), LISTENER-PORT= (bei PTYPE=SOCKET auch unter BS2000/OSD)

  Komponenten der Transportadresse eines fernen UPIC-Client oder einer TS-Anwendung, siehe Abschnitt „Adressinformationen bereitstellen“ auf Seite 150.

B

- USAGE=D (nur BS2000/OSD)

B
B

  USAGE=D legt fest, dass der Kommunikationspartner ein Dialog-Partner ist. Nachrichtenaustausch zwischen UTM-Anwendung und Client ist möglich.

- USP-HDR=

  Bei Socket-Anwendungen steuert dieser Parameter, für welche Ausgabe-Nachrichten openUTM einen Protokoll-Header aufbauen soll, siehe Abschnitt „USP-Header für Ausgabe-Nachrichten auf Socket-Verbindungen“ auf Seite 148.

BCAMAPPL-Anweisung auf Seite 289
Für UPIC-Clients und TS-Anwendungen können Sie damit zusätzliche Anwendungsnamen definieren.

- *appliname*

  Name der lokalen Anwendung, über den das Transportsystem die Verbindung zwischen Client und UTM-Anwendung aufbaut. Wird dieser Name für Socket-Anwendungen verwendet, dann darf er nicht von andersartigen Partnern benutzt werden.

- SIGNON-TAC=

  Gibt an, ob und wenn ja welcher Anmelde-Vorgang gestartet wird, wenn sich ein Client unter diesem Anwendungsnamen anmeldet, siehe Abschnitt „Anmelde-Vorgänge für Clients generieren“ auf Seite 141.

X/W

- TSEL-FORMAT=, LISTENER-ID= (nur unter Unix- und Windows-Systemen), LISTENER-PORT= (bei PTYPE=SOCKET auch unter BS2000/OSD), T-PROT=

  Komponenten der Transportadresse, unter der der Client die UTM-Anwendung anspricht, siehe Seite 148.

MAX-Anweisung auf Seite 368
Standard- und Maximalwerte, die bei der Kommunikation von Clients mit der UTM-Anwendung relevant sind, werden mit den folgenden Operanden festgelegt:

- CONN-USERS=

  Auslastung der Anwendung steuern. Der Operand legt die maximale Anzahl der Benutzer fest, die gleichzeitig mit der Anwendung arbeiten dürfen. Bei einer Anwendung, für die keine Benutzerkennungen generiert sind, wird mit CONN-USERS= die maximale Anzahl Clients festgelegt, die sich gleichzeitig über LTERM-Partner an die Anwendung anschließen dürfen.

- TRMSGLTH=

  Maximale Länge der physischen Nachrichten, die zwischen Clients und der UTM-Anwendung ausgetauscht werden können.

B

- LOCALE= (nur unter BS2000/OSD)

B
B
B
B
B
B
B

Definiert die Standard-Sprachumgebung (Locale) der UTM-Anwendung. Das hier generierte Locale wird den Clients, die sich über LTERM-Partner oder LTERM-Pools anschließen, als Standardwert für die Sprachumgebung zugeordnet. Die Standardeinstellung ist wirksam, solange für diese Objekte in den entsprechenden LTERM- oder TPOOL-Anweisungen kein eigenes Locale definiert wird. Sehen Sie dazu auch Abschnitt „Sprachumgebung festlegen - Locale definieren“ auf Seite 238.

5.1.2 LTERM-Pools

Für einen LTERM-Pool wird eine bestimmte Anzahl LTERM-Partner mit gleichen logischen Eigenschaften als logische Anschlusspunkte für Clients definiert. Über diese LTERM-Partner können sich unterschiedliche Clients mit gleichen technischen Eigenschaften (Partner- und Prozessortyp) dynamisch an eine UTM-Anwendung anschließen. Die Zuordnung besteht nur für die Dauer einer Session, es besteht keine statische Zuordnung zwischen Client und LTERM-Partner.

Ein LTERM-Pool muss in einer TPOOL-Anweisung konfiguriert werden (an Stelle von LTERM-/PTERM-Anweisungen). Zusätzlich kann noch wie beim Einzelanschluss eine BCAMAPPL-Anweisung nötig sein, siehe Seite 128. Ebenso gelten die Einstellungen in der MAX-Anweisung auch für LTERM-Pools, siehe Seite 128.

Es können verschiedene Arten von LTERM-Pools konfiguriert werden:

- LTERM-Pools, über die sich nur Clients eines bestimmten Typs (PTYPE=), die sich auf einem bestimmten Rechner (PRONAM=) befinden, anschließen können.
- LTERM-Pools, über die sich Clients eines bestimmten Typs anbinden können, unabhängig davon, an welchem Rechner sie sich befinden (Offene LTERM-Pools)

BB In UTM-Anwendungen unter BS2000/OSD können Sie darüber hinaus folgende Arten von LTERM-Pools generieren:

BB
- LTERM-Pools für alle Clients mit Benutzerdienstprotokoll unabhängig vom Typ, die sich auf einem bestimmten Rechner befinden.

BB
- LTERM-Pools für alle Clients mit Benutzerdienstprotokoll unabhängig von ihrem Typ und unabhängig von dem Rechner, auf dem sie sich befinden.

TPOOL-Anweisung auf Seite 518
Die wichtigsten Eigenschaften für LTERM-Pools werden mit den folgenden Operanden festgelegt:

- BCAMAPPL=

  Name der lokalen Anwendung, über den das Transportsystem die Verbindung zwischen Client und UTM-Anwendung aufbaut. Dieser Name muss in einer BCAMAPPL-Anweisung oder mit MAX ...APPLINAME= definiert sein. Wenn Sie diesen Operanden weglassen, wird der Name aus MAX ...APPLINAME= genommen.

BB Im BS2000 dürfen Clients mit Benutzerdienstprotokoll nur den Namen aus MAX ... APPLINAME= verwenden.

- CONNECT-MODE=

  Mit CONNECT-MODE= legen Sie fest, ob sich ein UPIC-Client oder eine TS-Anwendung mehrfach unter demselben Namen über den LTERM-Pool an die Anwendung anschließen kann.

- KSET=

  Keyset des LTERM-Pools, das über Keycodes die Zugriffsrechte der Clients festlegt, die sich über den LTERM-Pool an die UTM-Anwendung anschließen.

- USER-KSET=

  In UTM-Anwendungen mit Benutzerkennungen legt das Keyset USER-KSET für UPIC-Clients und TS-Anwendungen eingeschränkte Zugriffsrechte (gegenüber KSET) fest. Das Keyset in USER-KSET wird wirksam, wenn der Client beim Aufbau der Verbindung/Conversation bzw. im Anmelde-Vorgang keine Benutzerkennung an openUTM übergibt.

- LOCK=

  Zugangsschutz des LTERM-Pools, d.h. Lockcode, der für alle LTERM-Partner des Pools vergeben wird. Die Verbindung wird nur hergestellt, wenn der Client sich mit einer Benutzerkennung bei openUTM anmeldet, die in ihrem Keyset über den entsprechenden Keycode verfügt.

- ENCRYPTION-LEVEL=

  Für UPIC-Clients und unter BS2000/OSD zusätzlich für bestimmte Terminalemulationen legen Sie die minimale Verschlüsselungsebene fest, die auf Verbindungen zu Clients eingehalten werden muss. Sie können Clients, die sich über diesen LTERM-Pool anschließen, auch als vertrauenswürdig einstufen. Zur Verschlüsselung siehe auch Abschnitt „Nachrichten-Verschlüsselung auf Verbindungen zu Clients“ auf Seite 225.

- LTERM=

  LTERM-Präfix, aus dem mit *number* LTERM-Partner des LTERM-Pools eindeutige LTERM-Partnernamen erzeugt werden.

- NUMBER=

  Anzahl der für diesen LTERM-Pool konfigurierten LTERM-Partner. Damit ist implizit auch die maximale Anzahl der Clients festgelegt, die sich an diesen LTERM-Pool anschließen können.

- PRONAM=

  Name des Rechners, auf dem sich der Client befinden muss, wenn er sich über diesen LTERM-Pool anschließt.

B
B
- PROTOCOL=

  gibt an, ob mit oder ohne Benutzerdienstprotokoll gearbeitet wird.

- PTYPE=

  Typ des Clients, der sich über diesen LTERM-Pool anschließt.

B
- LOCALE=

B
B
B
B
  LTERM-spezifische Sprachumgebung, die für alle Clients gilt, die sich über LTERM-Partner dieses Pools an die Anwendung anschließen. Diese Sprachumgebung wird von openUTM auch für die Ausgabe von Meldungen verwendet, solange noch kein Benutzer angemeldet ist.

Zuordnung des Clients bei Anschluss über LTERM-Pool

Bei Clients, die sich über einen LTERM-Pool an eine Anwendung anschließen wollen, ist zu beachten, dass openUTM einen Client immer nur genau einem oder keinem LTERM-Pool zuordnet. Bei der Auswahl des LTERM-Pools bewertet openUTM dabei die Übereinstimmung im Rechnernamen höher als die Übereinstimmung des Client-Typs.

Die folgende Tabelle zeigt die Reihenfolge an, in der openUTM versucht, einen Client über die generierten PTERMs und LTERM-Pools anzuschließen. Die gerasterten Tabellenreihen stellen LTERM-Pools dar, die nur in einer UTM-Anwendung unter BS2000/OSD existieren können. Sie werden unter Unix- und Windows-Systemen übersprungen.

	Zuordnung des Clients		KDCDEF-Anweisungen: Definition des Clients		
	1.		PTERM	LTERM=ltermname PRONAM=processorname	
	2.		TPOOL	PTYPE=partnertyp PRONAM=processorname	
B B	3.		TPOOL	PTYPE=*ANY PRONAM=processorname	*nur in BS2000/OSD*
	4.		TPOOL	PTYPE=partnertyp PRONAM=*ANY	
B B	5.		TPOOL	PTYPE=*ANY PRONAM=*ANY	*nur in BS2000/OSD*

1. Beim Verbindungsaufbau wird zunächst überprüft, ob für den Client eine PTERM-Anweisung existiert. Ein Client, der mit einer PTERM-Anweisung explizit generiert wurde, kann sich nicht über einen LTERM-Pool an eine UTM-Anwendung anschließen.
2. Ein Client, für dessen Rechnernamen (PRONAM) und Typ (PTYPE) ein LTERM-Pool generiert ist, wird genau diesem LTERM-Pool oder keinem zugeordnet.

B B B 3. (nur in BS2000/OSD)
Wenn kein LTERM-Pool mit dem Rechnernamen und Typ des Clients existiert, wird der Client dem LTERM-Pool mit gleichem Rechnernamen und PTYPE=*ANY zugeordnet.

4. Wenn kein solcher LTERM-Pool existiert, wird der Client einem „offenen“ LTERM-Pool mit gleichem Typ und PRONAM=*ANY zugeordnet, d.h. alle Clients eines Typs können sich an die UTM-Anwendung anschließen, egal an welchem Rechner sie sich befinden.

B B B 5. (nur in BS2000/OSD)
Wenn auch kein solcher LTERM-Pool vorhanden ist, wird der Client einem LTERM-Pool mit PTYPE=*ANY und PRONAM=*ANY zugeordnet.

Ist auch kein solcher LTERM-Pool vorhanden, wird der Verbindungswunsch abgelehnt.

5.1.3 LTERM-Bündel

Mit einem LTERM-Bündel (Verbindungsbündel) verteilen Sie Asynchron-Nachrichten an eine logische Partner-Anwendung gleichmäßig auf mehrere parallele Verbindungen. Hinter der logischen Partner-Anwendung können sich auch mehrere Instanzen der Partner-Anwendungen (z.B. UTM-Cluster-Anwendung) verbergen. Ein solches Vorgehen ist dann sinnvoll, wenn eine UTM-Anwendung sehr viele asynchrone Nachrichten an eine Partner-Anwendung schickt und dadurch die Transportverbindung überlastet werden könnte.

Sie definieren ein LTERM-Bündel mit LTERM- und PTERM-Anweisungen, wie bereits im Abschnitt „Clients über LTERM-Partner anschließen“ auf Seite 125 dargestellt. Der folgende Text beschreibt die Punkte, die Sie für LTERM-Bündel zusätzlich beachten müssen.

Ein LTERM-Bündel besteht aus einem Master-LTERM und mehreren Slave-LTERMs. Die Slave-LTERMs, die einem PTERM mit PTYPE=APPLI oder PTYPE=SOCKET zugeordnet sein müssen, werden per Generierung einem Master-LTERM zugewiesen.

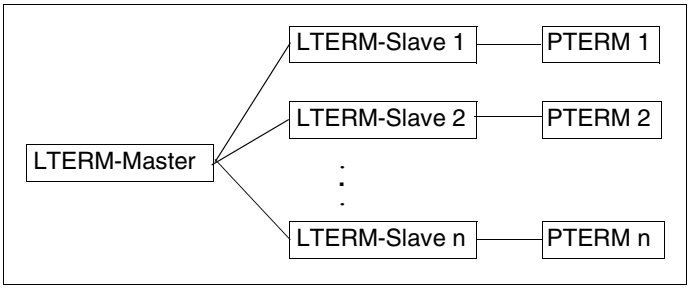


Bild 11: Beispiel LTERM-Bündel

FPUT-/DPUT-Aufrufe

FPUT- und DPUT-Aufrufe, die Teilprogramme an das Master-LTERM richten, werden beim Transaktionsende einem der Slave-LTERMs zugewiesen:

- openUTM versucht zuerst, ein Slave-LTERM zu finden, zu dessen PTERM eine Verbindung aufgebaut ist. Findet openUTM keine solche Verbindung, so wird ein Slave-LTERM gesucht, das mit RESTART=YES generiert ist.

  Findet openUTM ein Slave-LTERM, dann werden alle asynchronen Nachrichten, die in dieser Transaktion an dieses Master-LTERM gerichtet wurden, dem Slave-LTERM zugeordnet.

- Findet openUTM kein Slave-LTERM, werden alle mit FPUT oder DPUT an das Master-LTERM gerichteten Nachrichten verworfen.

- Ist ein Slave-LTERM mit RESTART=NO generiert und die Verbindung wird abgebaut oder geht verloren, werden alle Nachrichten verworfen, die an diesem LTERM zur Ausgabe anstehen.

Teilprogramme können FPUT- und DPUT-Aufrufe auch direkt an die Slave-LTERMs richten. Diese FPUTs unterliegen dann allerdings nicht dem oben beschriebenen Verteilalgorithmus.

Anzeige im KB-Kopf

Über die Slave-LTERMs eines LTERM-Bündels können auch Nachrichten empfangen werden. In Vorgängen, die für empfangene Nachrichten gestartet werden, zeigt openUTM im KB-Kopf immer den Namen des LTERM an, über das die Nachricht empfangen wurde. Für LTERM-Bündel gilt also:
In Vorgängen, die für Nachrichten gestartet werden, die über ein Slave-LTERM empfangen wurden, wird im KB-Kopf der Name dieses Slave-LTERMs angezeigt und **nicht** der Name des Master-LTERMs.

Mit Hilfe des KDCS-Aufrufs INIT PU können Sie sich darüber informieren, ob das LTERM im KB-Kopf Slave eines LTERM-Bündels ist und wie das Master-LTERM heißt (siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“).

LTERM-Anweisung auf Seite 354
Zusätzlich zu den bereits aufgeführten Eigenschaften für LTERM-Partner (siehe Seite 125), müssen für LTERM-Bündel folgende Operanden angegeben werden:

- BUNDLE=

  Gibt bei der Definition eines Slave-LTERM das zugehörige Master-LTERM an. Das hier angegebene Master-LTERM muss in einer vorangegangenen LTERM-Anweisung generiert worden sein:

```
LTERM master, ...
LTERM slave1, BUNDLE=master, ...
LTERM slave2, BUNDLE=master, ...

PTERM slave1, LTERM=slave1, PTYPE=APPLI|SOCKET, ...
PTERM slave2, LTERM=slave2, PTYPE=APPLI|SOCKET, ...
```

- RESTART=

  Bestimmt die Behandlung von Asynchron-Nachrichten bei Verbindungsabbau zum Client. Nachrichten, die an einem LTERM zur Ausgabe anstehen, das mit RESTART=NO generiert ist, werden ggf. verworfen (siehe Abschnitt „FPUT-/DPUT-Aufrufe“ auf Seite 133).

Alle LTERM-Parameter der Slave-LTERMs, mit Ausnahme von *ltermname*, USER, QAMSG, RESTART und STATUS müssen mit denen des Master-LTERMs übereinstimmen. Andernfalls werden sie von KDCDEF mit den Angaben beim Master-LTERM überschrieben. Dabei wird keine Meldung ausgegeben.

Beim Zuweisen der FPUT- und DPUT-Aufrufe zu einem Slave-LTERM am Transaktionsende werden die Einstellungen von QAMSG und RESTART am Slave-LTERM ausgewertet.

Alle Slave-LTERMs in einem LTERM-Bündel sollten identisch generiert werden. KDCDEF überprüft diese Angaben jedoch nicht.

PTERM-Anweisung auf Seite 440
Zusätzlich zu den bereits aufgeführten Eigenschaften für physische Clients (siehe Seite 126) müssen für die den Slave-LTERMs in einem LTERM-Bündel zugeordneten PTERMs folgende Operanden angegeben werden:

- PTYPE=APPLI | SOCKET

  Alle PTERMs eines LTERM-Bündels müssen mit PTYPE=APPLI oder PTYPE=SOCKET generiert werden. Für alle PTERMs eines LTERM-Bündels muss hier derselbe PTYPE angegeben werden.

B

- USAGE=D (nur BS2000/OSD)

B

Alle PTERMs eines LTERM-Bündels müssen mit USAGE=D generiert werden.

Alle PTERMs eines LTERM-Bündels sollten die gleiche bzw. eine gleichartige Partner-Anwendung adressieren. KDCDEF überprüft diese Angaben jedoch nicht.

5.1.4 LTERM-Gruppen

In einer LTERM-Gruppe ordnen Sie mehrere LTERMs einer Verbindung zu. Soll eine UTM-Anwendung asynchrone Nachrichten gemäß ihrer Zugehörigkeit zu verschiedenen Funktionsbereichen an unterschiedliche Partner-Anwendungen senden, muss jedem der Funktionsbereiche in den Partner-Anwendungen ein eigenes LTERM zugeordnet werden.

Die Teilprogramme richten ihre FPUT- und DPUT-Aufrufe abhängig von der Funktion an das passende LTERM. Ist die Zuordnung von Partner-Anwendung und Funktion 1 zu 1, wird jedem LTERM ein PTERM zugeordnet. Ist die Zuordnung von Partner-Anwendung und Funktion n zu 1 und wechselt diese Zuordnung ggf., werden n LTERMs einem PTERM zugeordnet.

Eine LTERM-Gruppe besteht aus einem oder mehreren Alias-LTERMs, den Gruppen-LTERMs, und einem Primary-LTERM. Die Gruppen-LTERMs definieren Sie mit LTERM-Anweisungen, wie im Abschnitt „Clients über LTERM-Partner anschließen“ auf Seite 125 dargestellt. Ein Gruppen-LTERM darf keinem PTERM zugeordnet werden.

Das Primary-LTERM muss entweder ein normales LTERM oder das Master-LTERM eines LTERM-Bündels sein. Ist das Primary-LTERM ein normales LTERM, muss ihm ein PTERM mit PTYPE=APPLI oder PTYPE=SOCKET zugeordnet sein. Sie definieren das Primary-LTERM wie im Abschnitt Abschnitt „Clients über LTERM-Partner anschließen“ auf Seite 125 dargestellt.

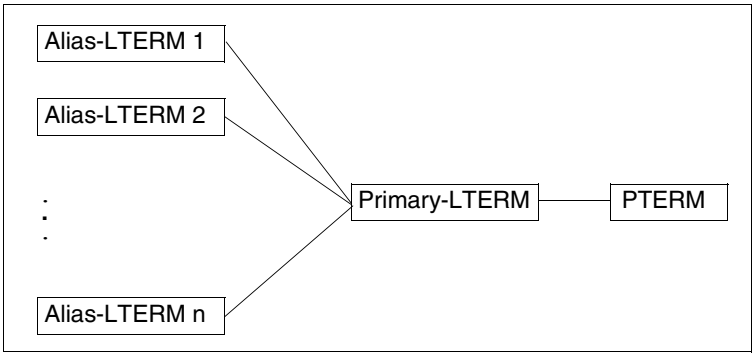


Bild 12: Beispiel LTERM-Gruppe

LTERM-Gruppen können auch in Verbindung mit LTERM-Bündeln eingesetzt werden. In diesem Fall ist das Primary-LTERM das Master-LTERM des LTERM-Bündels.

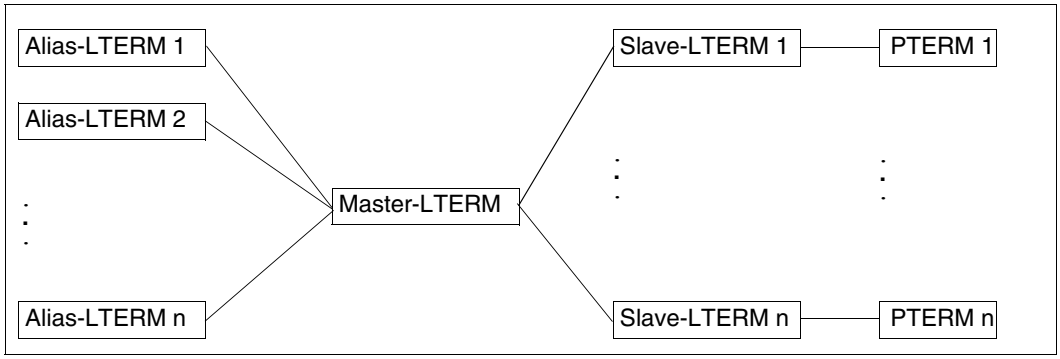


Bild 13: Beispiel LTERM-Gruppe in Verbindung mit einem LTERM-Bündel

FPUT-/DPUT-Aufrufe

FPUT- und DPUT-Aufrufe, die Teilprogramme an ein Alias-LTERM richten, werden wie folgt bearbeitet:

In einer LTERM-Gruppe ohne LTERM-Bündel:

FPUT- und DPUT-Aufrufe an ein Alias-LTERM werden von openUTM über das PTERM gesendet, das dem Primary-LTERM zugeordnet ist.

In einer LTERM-Gruppe, deren Primary-LTERM das Master-LTERM eines LTERM-Bündels ist:

Werden FPUT- und DPUT-Aufrufe an ein Alias-LTERM in einer solchen LTERM-Gruppe gerichtet, weist openUTM beim Transaktionsende alle asynchronen Nachrichten, die in dieser Transaktion an Alias-LTERMs der Gruppe gerichtet wurden, genau einem der Slave-LTERMs zu.

Dieses Vorgehen garantiert, dass beim Empfänger die Reihenfolge der Nachrichten erhalten bleibt, die in einer Transaktion für eine LTERM-Gruppe erzeugt wurden.

Teilprogramme können FPUT- und DPUT-Aufrufe auch direkt an das Primary-LTERM richten.

Anzeige im KB-Kopf

Wenn das Primary-LTERM einer LTERM-Gruppe kein Master-LTERM eines LTERM-Bündels ist, können über das Primary-LTERM auch Nachrichten empfangen werden. In Vorgängen, die für empfangene Nachrichten gestartet werden, zeigt openUTM im KB-Kopf immer den Namen des LTERM bzw. LPAP an, über das die Nachricht empfangen wurde. Für LTERM-Gruppen gilt also:

In Vorgängen, die für Nachrichten gestartet werden, die über das Primary-LTERM empfangen wurden, wird im KB-Kopf der Name des Primary-LTERMs angezeigt und **nicht** der Name eines Alias-LTERMs.

Mit Hilfe des KDCS-Aufrufs INIT PU können Sie sich darüber informieren, ob das LTERM im KB-Kopf Primary-LTERM einer LTERM-Gruppe ist (siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“).

LTERM-Anweisung auf Seite 354
Zusätzlich zu den bereits aufgeführten Eigenschaften für LTERM-Partner (siehe Seite 125), müssen für eine LTERM-Gruppe folgende Operanden angegeben werden:

- GROUP=

  Gibt bei der Definition eines Alias-LTERMs das zugehörige Primary-LTERM an. Das hier angegebene Primary-LTERM muss in einer vorangegangenen LTERM-Anweisung generiert worden sein:

```
LTERM primary, ...
PTERM primary, LTERM=primary, PTYPE=APPLI | SOCKET, ...

LTERM alias1, GROUP=primary, ...
LTERM alias2, GROUP=primary, ...
```

Alle LTERM-Parameter der Alias-LTERMs, mit Ausnahme von *ltermname*, USER und STATUS müssen mit denen des Primary-LTERMs übereinstimmen. Andernfalls werden sie von KDCDEF mit den Angaben beim Primary-LTERM überschrieben. Dabei wird keine Meldung ausgegeben.

Bei einem FPUT- oder DPUT-Aufruf werden nur die Generierungsparameter des Primary-LTERMs bewertet.

PTERM-Anweisung auf Seite 440
Zusätzlich zu den bereits aufgeführten Eigenschaften für physische Clients (siehe Seite 126) müssen für das einem Primary-LTERM einer LTERM-Gruppe zugeordnete PTERM folgende Operanden angegeben werden:

- PTYPE=APPLI | SOCKET

  Das PTERM in einer LTERM-Gruppe muss mit PTYPE=APPLI oder PTYPE=SOCKET generiert werden.

B

- USAGE=D (nur BS2000/OSD)

B

  Das PTERM in einer LTERM-Gruppe muss mit USAGE=D generiert werden.

5.1.5 OpenCPIC-Clients anschließen

OpenCPIC-Clients werden als OSI TP-Partner behandelt. Daher beschreibt dieser Abschnitt nur Client-spezifische Besonderheiten der OSI TP-Generierung.

Generierung

Ein OpenCPIC-Client wird ähnlich wie eine Server-Server-Kopplung generiert, siehe Abschnitt „Verteilte Verarbeitung über das OSI TP-Protokoll“ auf Seite 88. Der einzige Unterschied besteht darin, dass keine LTAC-Anweisungen nötig sind, wenn es sich um einen reinen Client handelt.

Die Anweisungen ABSTRACT-SYNTAX, APPLICATION-CONTEXT und TRANSFER-SYNTAX sind nur notwendig, wenn Sie einen eigenen Application Context definieren wollen.

Mehr Informationen zur Adressierung und zur Abstimmung mit der OpenCPIC-Generierung finden Sie ab Seite 157.

5.1.6 Anmeldeverfahren für Clients definieren

Dieser Abschnitt beschreibt die Schnittstelle zwischen Generierung und Anmeldeverfahren für Clients, wenn die Anwendung mit Benutzerkennungen generiert wird. Das Anmeldeverfahren setzt sich aus den beiden Schritten **Verbindungsaufbau** und **Anmelden** zusammen.

Der Verbindungsaufbau erfolgt über den Anwendungsnamen, der im Operanden BCAMAPPL bzw. bei OpenCPIC im Operanden LOCAL-ACCESS-POINT= angegeben ist.

Anmelden an eine UTM-Anwendung

Das Anmelden an eine UTM-Anwendung läuft über eine Benutzerkennung. Dazu sind folgende Schritte notwendig, unabhängig davon, ob das Standard-Anmeldeverfahren oder ein Anmelde-Vorgang verwendet wird:

- Bei Terminals muss der Terminal-Benutzer nach dem Aufbau der Verbindung seine Zugangsberechtigung nachweisen. Dazu muss er zumindest eine Benutzerkennung eingeben. Diese Benutzerkennung muss in einer USER-Anweisung generiert sein und wird auch **echte Benutzerkennung** genannt.
- TS-Anwendungen und UPIC-Clients werden nach dem Verbindungsaufbau unter einer so genannten **Verbindungs-Benutzerkennung** angemeldet. Diese Benutzerkennung ist eine von openUTM mit dem LTERM-Namen implizit erzeugte Benutzerkennung, falls in der LTERM-Anweisung im Operanden USER= keine Benutzerkennung angegeben ist. Wird im Operanden USER= eine Benutzerkennung angegegeben (explizite Verbindungs-Benutzerkennung), so muss diese mit einer USER-Anweisung generiert sein, siehe Abschnitt „Automatisches Anmelden unter einer bestimmten Benutzerkennung“ auf Seite 141. Diese Benutzerkennung kann dann nicht als echte Benutzerkennung verwendet werden.
- OpenCPIC-Clients werden nach dem Aufbau der Association unter ihrem **Association-Namen** angemeldet. Dieser wird zusammengesetzt aus dem im Operanden ASSOCIATION-NAMES= angegebenen Namen sowie der laufenden Nummer, z.B. ASSOC03, siehe Seite 429f.

UPIC-Clients, TS-Anwendungen und OpenCPIC-Clients können sich anschließend noch unter einer echten Benutzerkennung anmelden.

Den Ablauf des Anmeldeverfahren können Sie per Generierung beeinflussen, z.B. durch automatischen Verbindungsaufbau, automatisches Anmelden unter einer bestimmten Benutzerkennung, eigene Anmelde-Vorgänge oder indem Mehrfachanmeldung zugelassen wird.

Details zum Anmeldeverfahren finden Sie im jeweiligen openUTM-Handbuch „Einsatz von openUTM-Anwendungen“. Dort sind die einzelnen Schritte beschrieben, mit denen sich ein Client bei der UTM-Anwendung anmeldet.

5.1.6.1 Automatischer Verbindungsaufbau

Bestimmte Clients können so generiert werden, dass openUTM nach dem Anwendungsstart sofort versucht, die Verbindung zum Client aufzubauen. Dies ist möglich für:
– OpenCPIC-Clients,
– Einzeln generierte Terminals und TS-Anwendungen im BS2000/OSD
– Einzeln generierte TS-Anwendungen in Unix- und Windows-Systemen.

Der automatische Verbindungsaufbau wird wie folgt generiert:

- TS-Anwendungen und Terminals:

  ```
  PTERM ... ,CONNECT=YES
  ```

- OpenCPIC-Clients:

  ```
  OSI-LPAP ... ,CONNECT=n (n>0)
  ```

  Bei OpenCPIC-Clients muss auch der Client mit PARTAPPL ... ,CONNECT=n (n>0) generiert sein, siehe Handbuch „openUTM-Client für Trägersystem OpenCPIC“.

5.1.6.2 Automatisches Anmelden unter einer bestimmten Benutzerkennung

Sie können jedem explizit mit LTERM/PTERM definierten Client eine Benutzerkennung zuordnen, unter der dieser Client nach dem Verbindungsaufbau automatisch angemeldet wird (automatisches KDCSIGN). Für den Client gelten dann die Zugriffsrechte, die dieser Benutzerkennung zugeordnet sind, siehe Abschnitt „Securityfunktionen generieren“ auf Seite 145. Dazu sind folgende Generierungsanweisungen nötig:

```
LTERM ... USER=username
USER username ...
```

Terminals sind dann endgültig unter dieser Benutzerkennung angemeldet. Bei TS-Anwendungen und UPIC-Clients ist diese Benutzerkennung eine Verbindungs-Benutzerkennung und kann daher noch durch eine echte Benutzerkennung ersetzt werden, z.B. in einem Anmelde-Vorgang, siehe unten.

5.1.6.3 Anmelde-Vorgänge für Clients generieren

Für Terminals, UPIC-Clients und TS-Anwendungen können Sie eigene Anmelde-Vorgänge programmieren. Ein Anmelde-Vorgang ist an einen Anwendungsnamen gebunden. Damit können Sie jedem Anwendungsnamen einen eigenen Anmelde-Vorgang zuordnen. Anwendungsnamen definieren Sie mit MAX APPLINAME= oder in einer BCAMAPPL-Anweisung.

Meldet sich ein Client über einen bestimmten Anwendungsnamen an, dann wird der zu diesem Anwendungsnamen gehörige Anmelde-Vorgang gestartet. Der Anwendungsname, unter dem sich ein Client anmeldet, wird bei PTERM/TPOOL im Operanden BCAMAPPL festgelegt.

Anmelde-Vorgänge werden wie folgt generiert:

- Der Anmelde-Vorgang für den Standard-Anwendungsnamen (definiert in MAX ... APPLINAME) wird generiert mit:

  ```
  TAC KDCSGNTC, PROGRAM=signon-prog1
  PROGRAM signon-prog1 ...
  ```

  signon-prog1 ist der Name des Teilprogramms, das im Anmelde-Vorgang als erstes durchlaufen wird.

  Wird für den Standard-Anwendungsnamen ein Anmelde-Vorgang generiert, so ist dieser zugleich der Standardwert für alle mit BCAMAPPL generierten Anwendungsnamen.

- Der Anmelde-Vorgang für einen per BCAMAPPL definierten Anwendungsnamen wird generiert mit:

  ```
  BCAMAPPL applname2...,SIGNON=signon-tac
  TAC signon-tac, PROGRAM=signon-prog2
  PROGRAM signon-prog2
  ```

  signon-prog2 ist der Name des Teilprogramms, das im Anmelde-Vorgang als erstes durchlaufen wird.

- Sollen Anmelde-Vorgänge auch für UPIC-Clients durchlaufen werden, müssen Sie in der SIGNON-Anweisung Folgendes angeben:

  ```
  SIGNON ... UPIC=YES
  ```

  Ohne diese Angabe können UPIC-Clients Anmelde-Vorgänge nicht nutzen, auch dann nicht, wenn für den entsprechenden Anwendungsnamen ein Anmelde-Vorgang generiert ist.

Einzelheiten zur Programmierung finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“.

5.1.6.4 Mehrfachanmeldung

Unter einer Benutzerkennung kann sich zu einer Zeit normalerweise nur ein Client an die Anwendung anmelden, ein zweiter Anmeldeversuch unter der gleichen Benutzerkennung wird abgelehnt. Wenn Sie eine Mehrfachanmeldung für bestimmte Benutzerkennungen zulassen wollen, müssen Sie Folgendes generieren:

```
SIGNON ... MULTI-SIGNON=YES
```

Damit können sich unter einer echten Benutzerkennung ohne wiederanlauffähigen Vorgangskontext (USER *username*... RESTART=NO) zu einer Zeit verschiedene Clients an die Anwendung anmelden, davon jedoch nur **ein** Terminal-Client.

OpenCPIC-Clients, die die Functional Unit „Commit“ gewählt haben, können sich dann unter jeder echten Benutzerkennung mehrfach anmelden.

5.1.7 Maximale Wartezeiten für die Dialog-Führung festlegen

In der KDCDEF-Steueranweisung MAX können mit den Operanden TERMWAIT= und PGWTTIME= sowie dem Operanden IDLETIME= der PTERM-Anweisung maximale Wartezeiten für die Dialog-Führung festgelegt werden.

- Mit dem Operanden PGWTTIME= kann die Zeitspanne eingestellt werden, die zwischen der Ausgabe einer Dialog-Nachricht an den Client nach einem blockierenden Aufruf (z.B. einem PGWT-Aufruf) und der nachfolgenden Dialog-Eingabe maximal verstreichen darf. Erfolgt in diesem Zeitraum keine Eingabe, unterbricht openUTM den Vorgang zwangsweise.
- Mit dem Operanden TERMWAIT= kann die Zeitspanne eingestellt werden, die zwischen einer Dialog-Ausgabe an einem Terminal nach einem PEND KP und der nachfolgenden Dialog-Eingabe maximal verstreichen darf. Erfolgt in diesem Zeitraum keine Eingabe, unterbricht openUTM den Vorgang zwangsweise.
- Über den Operanden IDLETIME= kann man die Wartezeit nach PEND RE und PEND FI/ER/FR, also nach dem Transaktionsende, begrenzen.
  Die Überwachung der Wartezeit nach Transaktionsende dient dem Datenschutz. Falls ein Benutzer vergisst, sich nach Beendigung seiner Arbeit von der Anwendung abzumelden, kann man auf diese Weise das Risiko verringern, dass Unbefugte am Client arbeiten können **ohne** sich anzumelden.

Beachten Sie bitte, dass bei nicht blockierenden Aufrufen der UTM-Prozess freigegeben wird und andere Aufgaben übernehmen kann, während der Prozess bei blockierenden Aufrufen belegt bleibt. Wartet ein Vorgang nach einem PEND KP oder einem PGWT KP-Aufruf auf eine Dialog-Eingabe, dann bleibt die Transaktion offen („Mehrschritt-Transaktion“) und hält in der Regel globale Betriebsmittel gesperrt (z.B. GSSBs, Datenbereiche in DB-Systemen). Eine Verzögerung bei der Dialog-Eingabe würde die Arbeit anderer Benutzer blockieren, die ebenfalls auf diese Daten zugreifen wollen. Eine Zeitüberwachung für die genannten Fälle verhindert solche unerwünschten Effekte.

Eine Anleitung für den Benutzer muss unbedingt auf diesen Zusammenhang hinweisen. Sie muss ihm die kritischen Punkte in der Dialog-Führung zeigen, wo er die Performance der ganzen Anwendung beeinträchtigt, wenn er die Eingabe verzögert.

5.1.8 Securityfunktionen generieren

Die Securityfunktionen bestehen aus folgenden Komponenten:

- Zugangsschutz:
  Den Zugangsschutz definieren Sie in der USER-Anweisung, siehe unten.
- Administrationsberechtigung:
  Die Administrationsberechtigung vergeben Sie in der USER-Anweisung bzw. in der OSI-LPAP-Anweisung (OpenCPIC), siehe Seite 146.
- Zugriffsschutz:
  Den Zugriffsschutz legen Sie mit den Operanden KSET, USER-KSET bzw. ASS-KSET der USER-, LTERM-, TPOOL- bzw. OSI-LPAP-Anweisung fest. Der Zugriffsschutz muss im Rahmen des Lock/Keycode-Konzepts bzw. des Access-List-Konzepts definiert werden und ist ausführlich in Abschnitt „Zugriffskontrolle“ auf Seite 216 beschrieben. Der Zugriffsschutz bei OpenCPIC-Clients wird so generiert wie im Abschnitt „Schutzmaßnahmen in der Auftragnehmer-Anwendung“ auf Seite 224 beschrieben.
- Verschlüsselung:
  Die Verschlüsselungsebene wird im Operanden ENCRYPTION-LEVEL in der PTERM-, TPOOL- bzw. TAC-Anweisung festgelegt. Die ausführliche Beschreibung der Nachrichtenverschlüsselung finden Sie in Abschnitt „Nachrichten-Verschlüsselung auf Verbindungen zu Clients“ auf Seite 225f. Verschlüsselung durch openUTM wird für OpenCPIC-Clients nicht unterstützt.

5.1.8.1 Zugangsschutz definieren

Der Zugangsschutz ist nur für echte Benutzerkennungen relevant und wird in der USER-Anweisung generiert.

Sie definieren den Zugangsschutz, indem Sie der Benutzerkennung ein Passwort zuordnen und für das Passwort eine bestimmte Komplexitätsstufe festlegen. Das Passwort darf für die Client/Server-Kommunikation nicht hexadezimal angegeben werden.

USER *userid-name*,PASS=*password*,PROTECT-PW=*complexity-level*

Der Client muss beim Anmelden die für *userid-name* und für *password* angegebenen Werte an openUTM übergeben.

5.1.8.2 Administrationsberechtigungen vergeben

- Für die Benutzerkennung können Sie in der USER-Anweisung die Administrationsberechtigung vergeben:

  ```
  USER userid-name,PERMIT=ADMIN
  ```

B B B Unter BS2000/OSD können Sie für die Benutzerkennung darüber hinaus UTM-SAT-Administrationsberechtigung vergeben (Operand PERMIT) und Art und Umfang der SAT-Protokollierung festlegen (Operand SATSEL):

B
```
USER userid-name,PERMIT=SATADM,SATSEL=...
```

- Für einen OpenCPIC-Client können Sie die Administrationsberechtigung bei OSI-LPAP vergeben:

  ```
  OSI-LPAP ... PERMIT=ADMIN
  ```

B B Unter BS2000/OSD können Sie für den Client darüber hinaus UTM-SAT-Administrationsberechtigung vergeben:

B OSI-LPAP ... PERMIT=SATADM bzw. PERMIT=(ADMIN,SATADM)

Meldet sich der OpenCPIC-Client unter einer echten Benutzerkennung an, so gelten die Zugriffsrechte, die für die Benutzerkennung generiert sind, und nicht die Zugriffsrechte des OSI-LPAP.

5.1.9 Wiederanlauf generieren

Die Wiederanlauffunktion für einen Client ist an die Benutzerkennung gekoppelt, unter der sich der Client an die UTM-Anwendung anmeldet.

Wiederanlauffunktion für echte Benutzerkennungen

Die Wiederanlauffunktion für echte Benutzerkennungen legen Sie im Operanden RESTART der USER-Anweisung fest.

```
USER userid-name ... RESTART=YES | NO
```

Wenn RESTART=YES generiert ist, dann spielen für einen Vorgangswiederanlauf noch der Typ des Client und eventuell generierte Anmelde-Vorgänge eine Rolle:

– Ist für den Client, der sich unter dieser Benutzerkennung anmeldet, ein Anmelde-Vorgang generiert, dann steuert dieser, ob der Vorgangswiederanlauf durchgeführt oder der offene Vorgang abnormal beendet wird, siehe Beschreibung des Anmelde-Vorgangs im openUTM-Handbuch „Anwendungen programmieren mit KDCS“.

– Wenn ein Terminal oder eine TS-Anwendung sich nicht über einen Anmelde-Vorgang anmeldet, dann veranlasst openUTM immer einen Vorgangswiederanlauf.

– Wenn ein UPIC-Client sich nicht über einen Anmelde-Vorgang anmeldet, dann muss der UPIC-Client den Vorgangswiederanlauf explizit initiieren, ansonsten wird der offene Vorgang abnormal beendet, siehe Handbuch „openUTM-Client für das Trägersystem UPIC“.

– Bei OpenCPIC-Clients ist Wiederanlauf nur bei Cooperative Processing möglich (Functional Unit ungleich „Commit“). Der OpenCPIC-Client muss dann den Vorgangswiederanlauf explizit initiieren, ansonsten wird der offene Vorgang abnormal beendet, siehe Handbuch „openUTM-Client für Trägersystem OpenCPIC“.

Wiederanlauffunktion für Verbindungs-Benutzerkennungen

Falls sich einzeln generierte TS-Anwendungen über implizite (von openUTM erzeugte) Verbindungs-Benutzerkennungen anmelden, dann wird die Wiederanlauffunktion über den Operanden RESTART in der LTERM-Anweisung gesteuert:

```
LTERM ltermname ... RESTART=YES | NO
```

Dieser Parameter ist für den Vorgangswiederanlauf irrelevant, wenn die TS-Anwendung über eine explizit generierte Verbindungs-Benutzerkennung oder eine echte Benutzerkennung angemeldet wird.

5.1.10 USP-Header für Ausgabe-Nachrichten auf Socket-Verbindungen

Für die Kommunikation der UTM-Anwendung mit TS-Anwendungen über die Socket-Schnittstelle wird oberhalb von TCP/IP ein UTM-Socket-Protokoll (USP) verwendet. Mit Hilfe dieses Protokolls kann openUTM einen über die Socket-Schnittstelle empfangenen Bytestream in eine Nachricht umsetzen. Die Partner-Anwendung muss das Protokoll versorgen und der Eingabe-Nachricht als Protokoll-Header voranstellen. Bei Ausgabe-Nachrichten erzeugt openUTM standardmäßig kein Protokoll.

Per Generierungsoption lässt sich einstellen, dass openUTM auch bei Ausgabe-Nachrichten einen Protokoll-Header voranstellt. Dies legen Sie in der PTERM oder TPOOL-Anweisung mit dem Operanden USP-HDR= fest:

- USP-HDR=ALL bewirkt, dass openUTM jeder Ausgabe-Nachricht (Dialog- oder Asynchron-Nachricht, K-Meldung) auf dieser Verbindung einen Protokoll-Header voranstellt.
- Bei USP-HDR=MSG wird der Protokoll-Header nur für K-Meldungen vorangestellt.
- USP-HDR=NO bedeutet keinen Protokoll-Header für Ausgabe-Nachrichten.

Der Aufbau des Protokoll-Headers ist im openUTM-Handbuch „Anwendungen programmieren mit KDCS“ beschrieben.

5.1.11 Code-Konvertierung

Bei der Kommunikation zwischen der UTM-Anwendung und einem Client kann es vorkommen, dass die beiden Kommunikationspartner mit unterschiedlichen Codes arbeiten (EBCDIC, ASCII/ISO8859-1). Damit die Kommunikation zwischen den Partnern trotzdem einfach ist, können Sie per Generierung für folgende Clients eine automatische Code-Konvertierung zwischen ASCII/ISO8859-1 und EBCDIC veranlassen:

B
- BS2000/OSD: TS-Anwendungen vom Typ SOCKET

X/W
- Unix-, Windows-Systeme:

X/W
  – OpenCPIC-Clients und TS-Anwendungen vom Typ APPLI und SOCKET

X/W
  – Server-Server-Kommunikation mit LU6.1- und OSI TP-Partnern

Dabei ist jedoch zu beachten, dass nur abdruckbare Nachrichten ausgetauscht werden, da Binärdaten durch eine Code-Umsetzung evtl. verfälscht werden.

Die Code-Konvertierung regeln Sie über den Operanden MAP=:

B
- BS2000/OSD:

B
```
PTERM/TPOOL ... MAP= USER | SYSTEM | SYS1 | SYS2 | SYS3 | SYS4
```

B
B
Unter BS2000/OSD können Sie verschiedene Umsetztabellen verwenden. Der Betreiber einer UTM-Anwendung unter BS2000/OSD kann auch eigene Tabellen erstellen.

X/W
- Unix-, Windows-Systeme:

X/W
```
PTERM/TPOOL ... MAP = USER | SYSTEM     (TS-Anwendungen)
```
X/W
```
OSI-CON ... MAP = USER | SYSTEM   (OSI TP-Partner)
```
X/W
```
SESCHA  ... MAP = USER | SYSTEM  (LU6.1-Partner)
```

X/W
Unter Unix- und Windows-Systemen kann die Umsetztabelle modifiziert werden.

Standardmäßig wird ohne Codeumsetzung gearbeitet, d.h. MAP=USER gesetzt. Bei MAP=SYSTEM bzw. SYS1, ..., SYS4 konvertiert openUTM die Daten zwischen ASCII und EBCDIC um. Zur Code-Konvertierung werden Umsetztabellen zur Verfügung gestellt.

Weitere Informationen zur Code-Konvertierung finden Sie im Anhang auf Seite 609.

5.1.12 Adressinformationen bereitstellen

Für TS-Anwendungen, ferne UPIC-Clients und OpenCPIC-Clients werden für den Verbindungsaufbau Adressinformationen benötigt. Diese werden in der UTM-Generierung hinterlegt. Bei Socket-Anwendungen gibt es keine Unterschiede zwischen BS2000/OSD und Unix- oder Windows-Systemen. Bei anderen Clients wirken sich dagegen die Eigenschaften des Transportsystems aus. Daher werden die Informationen in getrennten Abschnitten behandelt.

Portnummer 102 bei TCP/IP-Verbindungen

Bei Verbindungen über TCP/IP werden die Anwendungen über Portnummern angesprochen, wobei die Portnummer 102 eine besondere Rolle spielt. Sie müssen deshalb Folgendes beachten:

– In BS2000/OSD wird häufig die Portnummer 102 verwendet, u.a. deshalb, weil für Portnummer 102 kein BCMAP-Eintrag notwendig ist.

– In Unix-Systemen dagegen sollte die Portnummer 102 nur mit Vorsicht, in Windows-Systemen sollte die Portnummer 102 generell nicht verwendet werden. Der Grund liegt darin, dass Port 102 dort häufig als privilegierter Port verwendet wird oder nur von einem einzigen Prozess auf dem Rechner genutzt werden kann.

5.1.12.1 Adressinformationen für Clients vom Typ SOCKET bereitstellen

Bei der Kommunikation mit TS-Anwendungen über TCP/IP wird direkt die Socket-Schnittstelle verwendet.

Die für die Kommunikation benötigten Adressinformationen werden größtenteils in der UTM-Generierung bereitgestellt. Die restlichen Informationen ermittelt openUTM aus der Hosts-Datei/-Datenbank. Es sind deshalb auch auf BS2000/OSD keine BCMAP-Einträge nötig.

KDCDEF-Generierung

Die Adressinformationen werden in den Operanden LISTENER-PORT=, T-PROT= und PRONAM= der BCAMAPPL- und TPOOL/PTERM-Anweisung hinterlegt.

- BCAMAPPL-Anweisung

  Sie müssen für Socket-Anwendungen immer eine BCAMAPPL-Anweisung geben. Dabei gilt:

  – Bei LISTENER-PORT= müssen Sie eine Portnummer eintragen, unter der die UTM-Anwendung auf Anforderungen der Socket-Anwendung wartet. Die Portnummer muss in jedem Fall mit dem Kommunikationspartner abgestimmt sein.

  – Bei T-PROT= müssen Sie immer SOCKET angeben.

X/W
X/W
X/W

  – LISTENER-ID= ordnet der Verbindung wahlweise eine Listener-ID zu. Die Werte für LISTENER-ID von Nicht-Socket-Verbindungen und von Socket-Verbindungen können unabhängig voneinander vergeben werden.

- PTERM-Anweisung

  Wenn Sie die Socket-Anwendung einzeln generieren, dann geben Sie folgende Operanden und Parameter an:

  – Wenn die Socket-Anwendung die Verbindung aufbaut, dann muss der PTERM-Name das Format PRT*nnnnn* haben, *nnnnn* ist die Portnummer, von der die Socket-Anwendung die Verbindung aufbaut, eventuell ergänzt um führende Nullen.

  – Bei BCAMAPPL= geben Sie den in der BCAMAPPL-Anweisung definierten Anwendungsnamen ein.

  – Bei PRONAM= müssen Sie immer den ersten TCP/IP-Hostnamen angeben, der in der Datenbasis (Hosts-Datei, DNS, ...) generiert ist. Falls Sie die Funktion „Mapped Hostnamen“ verwenden wollen, müssen Sie hier den UTM-Hostnamen angeben (siehe Seite 118). Weitere in der Datenbasis (Hosts-Datei, DNS,...) angegebene Namen, sogenannte Aliasnamen, dürfen Sie hier nicht angeben.

  – Im Operanden LISTENER-PORT= müssen Sie die Portnummer angeben, an der die Socket-Anwendung auf Verbindungsaufbauwünsche wartet.

- TPOOL-Anweisung

  Wenn Sie die Socket-Anwendung über einen LTERM-Pool anbinden, dann gilt:

  – Bei PRONAM= müssen Sie immer den ersten TCP/IP-Hostnamen angeben, der in der Datenbasis (Hosts-Datei, DNS, ...) generiert ist. Falls Sie die Funktion „Mapped Hostnamen“ verwenden wollen, müssen Sie hier den UTM-Hostnamen angeben (siehe Seite 118). Weitere in der Datenbasis (Hosts-Datei, DNS,...) angegebene Namen, sogenannte Aliasnamen, dürfen Sie hier nicht angeben.

X/W
X/W
X/W

  Wenn Sie PRONAM=*ANY eintragen, dann dürfen sich die Clients von beliebigen Rechnern aus anmelden, vorausgesetzt, sie entsprechen dem Typ, der in PTYPE= angegeben wurde.

– Bei BCAMAPPL= geben Sie den in der BCAMAPPL-Anweisung definierten Anwendungsnamen ein.

Beispiel

In folgendem Beispiel wird lokal die Portnummer 10100 verwendet. Die Socket-Anwendung wird über LTERM-Pool angebunden und läuft im Rechner PCSOCK01.

```
BCAMAPPL BSPSOCK -
    ,LISTENER-PORT=10100 -
    ,T-PROT=SOCKET -
    ,...
TPOOL ...
    ,PTYPE=SOCKET -
    ,BCAMAPPL=BSPSOCK -
    ,PRONAM=PCSOCK01 -
    ,...
```

5.1.12.2 Adressinformationen für Clients vom Typ UPIC und APPLI in BS2000/OSD bereitstellen

B B B Für RFC1006- (bzw. ISO-) Verbindungen muss immer ein eigener Anwendungsname generiert werden. Dieser kann unabhängig vom Client-Typ für alle RFC1006- (bzw. ISO-) Verbindungen genutzt werden.

B **KDCDEF-Generierung**

B • BCAMAPPL-Anweisung

B B Bei T-PROT= geben Sie ISO oder RFC1006 an (diese beiden Werte werden im BS2000/OSD gleich behandelt). T-PROT=SOCKET dürfen Sie **nicht** angeben.

B • PTERM-Anweisung

B B Wenn Sie die Clients einzeln generieren, dann geben Sie folgende Operanden und Parameter an:

B B – als PTERM-Namen müssen Sie den Namen angeben, der bei der Generierung des Netzes für diesen Client definiert wurde.

B B – Bei BCAMAPPL= geben Sie den in der BCAMAPPL-Anweisung definierten Anwendungsnamen ein.

B B – Bei PRONAM= geben Sie den Namen des Rechners an, auf dem der Client läuft. Dieser Name wird bei der Generierung des Netzes festgelegt.

B B – Bei PTYPE= müssen Sie entweder UPIC-R oder APPLI angeben, PTYPE=*ANY ist nicht erlaubt.

B
- TPOOL-Anweisung

B
Wenn Sie Clients über LTERM-Pool anbinden, dann gilt:

B B
– Bei BCAMAPPL= geben Sie den in der BCAMAPPL-Anweisung definierten Anwendungsnamen ein.

B B B B
– Bei PRONAM= können Sie den Namen des Rechners angeben, auf dem die Clients laufen. Wenn Sie PRONAM=*ANY eintragen, dann dürfen sich die Clients von beliebigen Rechnern aus anmelden, vorausgesetzt, sie entsprechen dem Typ, der in PTYPE= angegeben wurde.

B B
– Bei PTYPE= müssen Sie entweder UPIC-R oder APPLI angeben, PTYPE=*ANY ist nicht erlaubt.

B **Abstimmung mit der BCAM-Generierung**

B B
Wird ein Client über RFC1006 an die UTM-Anwendung angebunden und verwendet eine Portnummer ≠ 102, dann ist für diesen Client ein BCMAP-Eintrag notwendig:

```
B  /BCMAP FUNCT=DEFINE,SUBFUNCT=GLOBAL,NAME=name-des-client, -
B  /       ES=partner-processorname,PPORT#=listener-port-nr,                -
B  /       PTSEL-I=tselektor-des-client
```

5.1.12.3 Adressinformationen für Clients vom Typ UPIC und APPLI in Unix- und Windows-Systemen bereitstellen

X/W X/W X/W
Auf Unix- und Windows-Systemen kann für den mit MAX APPLINAME= generierten Anwendungsnamen eine eigene BCAMAPPL-Anweisung geschrieben werden, mit der alle Parameter wie gewünscht festgelegt werden können.

X/W X/W X/W
Die Anbindung von Clients über RFC1006 ist im Abschnitt „Adressinformationen für das Transportsystem CMX bereitstellen (Unix- und Windows-Systeme)“ auf Seite 105 beschrieben. Die KDCDEF-Generierung muss alle notwendigen Adressinformationen enthalten.

X/W **KDCDEF-Generierung für RFC1006**

X/W
- BCAMAPPL-Anweisung

X/W X/W
  - *appliname*: Der Anwendungsname kann beliebig gewählt werden, muss jedoch netzweit eindeutig sein, da KDCDEF daraus einen T-Selektor erzeugt.

X/W X/W X/W X/W X/W X/W
  - Bei LISTENER-PORT= sollten Sie immer eine Portnummer eintragen. Die Portnummer muss in jedem Fall mit dem Kommunikationspartner abgestimmt sein. Wenn Sie bei LISTENER-PORT= keine Angabe machen, dann wird abhängig von der OPTION-Anweisung folgender Standardwert gesetzt:
    - Portnummer 102, wenn OPTION CHECK-RFC1006=YES angegeben wurde
    - die ungültige Portnummer 0, wenn CHECK-RFC1006=NO angegeben wurde

X/W
  - Bei T-PROT **müssen** Sie immer RFC1006 angeben.

X/W X/W X/W
  - Bei TSEL-FORMAT= geben Sie den Formatindikator für den Namen an, den Sie als *appliname* (s.o.) definiert haben. Es wird empfohlen, den Operanden TSEL-FORMAT= immer zu versorgen.

X/W
- PTERM-Anweisung

X/W X/W
  Wenn Sie den Client einzeln über PTERM-Anweisung generieren, dann geben Sie Folgendes an:

X/W X/W
  - als PTERM-Namen den T-Selektor des Client. Mit diesem T-Selektor muss der Client auf dem Client-Rechner als lokale Anwendung eingetragen sein.

X/W
  - Bei BCAMAPPL= geben Sie den oben definierten Anwendungsnamen ein.

X/W X/W
  - Bei LISTENER-PORT= geben Sie die Portnummer an, die der Client auf dem Client-Rechner als Ausgangsport verwendet.

X/W X/W X/W X/W X/W X/W
  - Bei PRONAM= müssen Sie immer den ersten TCP/IP-Hostnamen des Client-Rechners angeben, der in der Datenbasis (Hosts-Datei, DNS, ...) generiert ist. Falls Sie die Funktion „Mapped Hostnamen“ verwenden wollen, müssen Sie hier den UTM-Hostnamen angeben (siehe Seite 118). Weitere in der Datenbasis (Hosts-Datei, DNS,...) angegebene Namen, sogenannte Aliasnamen, dürfen Sie hier nicht angeben.

X/W
  - Bei PTYPE= müssen Sie entweder UPIC-R oder APPLI angeben.

X/W
- TPOOL-Anweisung

X/W Wenn Sie den Client über LTERM-Pool anschließen, dann geben Sie Folgendes an:

X/W – Bei BCAMAPPL= geben Sie den oben definierten Anwendungsnamen ein.

X/W – Bei PRONAM= können Sie den Namen des Rechners angeben, auf dem der Client läuft. Dies muss der erste TCP/IP-Hostname sein, der in der Datenbasis (Hosts-Datei, DNS,...) eingetragen ist. Falls Sie die Funktion „Mapped Hostnamen“ verwenden wollen, müssen Sie hier den UTM-Hostnamen angeben (siehe Seite 118). Weitere in der Datenbasis (Hosts-Datei, DNS,...) angegebene Namen, sogenannte Aliasnamen, dürfen Sie hier nicht angeben.

X/W Wenn Sie PRONAM=*ANY eintragen, dann dürfen sich die Clients von beliebigen Rechnern aus anmelden, vorausgesetzt, sie entsprechen dem Typ, der in PTYPE= angegeben wurde.

X/W – Bei PTYPE= müssen Sie entweder UPIC-R oder APPLI angeben.

X/W *Beispiel*

X/W In folgendem Beispiel wird lokal die Portnummer `10030` und in der fernen Anwendung die Portnummer `12030` verwendet. Der UPIC-Client läuft im Rechner `PCUPR`.

```
X/W BCAMAPPL BSPUPR -
X/W ,LISTENER-PORT=10030 -
X/W ,T-PROT=RFC1006 -
X/W ,T-SEL-FORMAT=T -
X/W ,...
X/W
X/W PTERM UPRPART -
X/W ,PTYPE=UPIC-R -
X/W ,BCAMAPPL=BSPUPR -
X/W ,PRONAM=PCUPR -
X/W ,LISTENER-PORT=12030 -
X/W ,T-PROT=RFC1006 -
X/W ,T-SEL-FORMAT=T -
X/W ,...
```

X/W Die Anweisungen für eine TS-Anwendung sind analog zu erstellen, nur dass bei PTERM PTYPE=APPLI anzugeben ist.

5.1.12.4 Besonderheiten bei LTERM-Pools auf Unix- und Windows-Systemen

X/W Über einen LTERM-Pool kann jede beliebige Partner-Anwendung auf einem bestimmten Rechner eine Verbindung zur UTM-Anwendung aufbauen, wenn die Partner-Anwendung vom richtigen Typ ist (PTYPE). Wird darüberhinaus in der TPOOL-Anweisung PRONAM=*ANY generiert, kann sich die TS-Anwendungen des generierten Typs von jedem beliebigen Rechner aus an die UTM-Anwendung anschließen.

X/W In der TPOOL-Anweisung wird kein Name (Stationsname) für diese Kommunikationspartner festgelegt. Die UTM-Anwendung ermittelt diesen Namen mit Hilfe der Transportadresse der TS-Anwendung, die die TS-Anwendung beim Verbindungsaufbau mitgeliefert hat.

X/W Dabei wird wie folgt vorgegangen:

- X/W Bei der Kommunikation einer TS-Anwendung mit der UTM-Anwendung wird versucht, über den lokalen Name Service den Rechnernamen herauszufinden.
- X/W Kann kein Rechnername herausgefunden werden, wird vom Netzprozess der Name *ANY vergeben.
- X/W Es wird versucht, den T-Selektor aus der Transportadresse zu gewinnen. Wird ein T-Selektor gefunden, dann wird er als Stationsname verwendet.
- X/W Kann kein T-Selektor gefunden werden, dann wird der Stationsname 'NETM*nnnn*' für die TS-Anwendung verwendet. *nnnn* steht für 0000 bis 9999 und wird von openUTM automatisch hochgezählt.

X/W Es ist günstig, LTERM-Pools über TCP/IP-Verbindungen anzusprechen, wenn der LTERM-Pool mit Prozessornamen (TPOOL ...,PRONAM=) generiert ist.

5.1.12.5 Adressinformationen für OpenCPIC-Clients in BS2000/OSD bereitstellen

B Die Adressinformationen für OpenCPIC-Clients werden in der ACCESS-POINT- und der
B OSI-CON-Anweisung hinterlegt. Diese Informationen müssen auch mit der OpenCPIC-Ge-
B nerierung abgestimmt sein.

B **KDCDEF-Generierung**

B
- ACCESS-POINT-Anweisung

B
B
  - Bei *access_point_name* definieren Sie den Namen des lokalen OSI TP-Zugriffspunktes.

B
B
  - Bei TRANSPORT-SELECTOR= müssen Sie den BCAM-Namen der lokalen UTM-Anwendung angeben.

B
B
B
B
  - Sofern benötigt, müssen Sie bei SESSION-SELECTOR= und PRESENTATION-SELECTOR= den Session- bzw. Presentation-Selektor des lokalen OSI TP-Zugriffspunktes eintragen. Werden diese Selektoren nicht benötigt, geben Sie jeweils *NONE an.

B
- OSI-CON-Anweisung

B
B
  - Bei LOCAL-ACCESS-POINT= tragen Sie den oben definierten Name des OSI TP-Zugriffspunktes ein.

B
B
  - Bei NETWORK-SELECTOR= müssen Sie den Namen des Rechners angeben, auf dem der OpenCPIC-Client läuft.

B
B
  - In TRANSPORT-SELECTOR= müssen Sie den BCAM-Anwendungsnamen des OpenCPIC-Clients eintragen.

B
B
B
  - Sofern benötigt, müssen Sie bei SESSION-SELECTOR= und PRESENTATION-SELECTOR= den Session- bzw. Presentation-Selektor des OpenCPIC-Client eintragen. Werden diese Selektoren nicht benötigt, geben Sie jeweils *NONE an.

B **Abstimmung mit der BCAM-Generierung**

B
B
Wird ein OpenCPIC-Client über RFC1006 an die UTM-Anwendung angebunden und verwendet eine Portnummer ≠ 102, dann ist für diesen Client ein BCMAP-Eintrag notwendig:

```
B  /BCMAP FUNCT=DEFINE,SUBFUNCT=GLOBAL,NAME=name-des-client, -
B  /       ES=partner-processorname,PPORT#=listener-port-nr,          -
B  /       PTSEL-I=tselektor-des-client
```

5.1.12.6 Adressinformationen für OpenCPIC-Clients in Unix- und Windows-Systemen bereitstellen

X/W Die Adressinformationen für OpenCPIC-Clients werden in der ACCESS-POINT- und der OSI-CON-Anweisung hinterlegt. Diese Informationen müssen auch mit der OpenCPIC-Generierung abgestimmt sein.

X/W Die Anbindung von Clients über RFC1006 ist im Abschnitt „Adressinformationen für das Transportsystem CMX bereitstellen (Unix- und Windows-Systeme)“ auf Seite 105 beschrieben. Die KDCDEF-Generierung muss alle notwendigen Adressinformationen enthalten.

X/W **KDCDEF-Generierung für RFC1006**

X/W Die KDCDEF-Generierung muss mit der Client-Generierung abgestimmt werden, siehe auch das Beispiel 3 auf Seite 162.

X/W
- ACCESS-POINT-Anweisung
  - X/W Bei *access_point_name* definieren Sie den Namen des lokalen OSI TP-Zugriffspunktes.
  - X/W Bei TRANSPORT-SELECTOR= müssen Sie den T-Selektor des lokalen OSI TP-Zugriffspunktes angeben. Dieser Name muss mit der Generierung auf der Client-Seite abgestimmt werden.
  - X/W Bei TSEL-FORMAT= geben Sie den Formatindikator des T-Selektors an. Es wird empfohlen, den Operanden TSEL-FORMAT= immer zu versorgen.
  - X/W Sofern benötigt, müssen Sie bei SESSION-SELECTOR= und PRESENTATION-SELECTOR= den Session- bzw. Presentation-Selektor des lokalen OSI TP-Zugriffspunktes eintragen. Diese Namen müssen mit der Generierung auf der Client-Seite abgestimmt werden.

    X/W Werden diese Selektoren nicht benötigt, müssen Sie jeweils *NONE angeben.
  - X/W Bei LISTENER-PORT= geben Sie die Portnummer an, unter der der OpenCPIC-Client die lokale UTM-Anwendung erreicht.
  - X/W Bei T-PROT= müssen Sie immer RFC1006 angeben.

X/W
- OSI-CON-Anweisung

X/W X/W
  – Bei LOCAL-ACCESS-POINT= tragen Sie den oben definierten Namen des OSI TP-Zugriffspunktes ein.

X/W X/W X/W X/W X/W X/W
  – Bei NETWORK-SELECTOR= müssen Sie den Namen des Rechners angeben, auf dem der OpenCPIC-Client läuft. Dies muss immer der erste TCP/IP-Hostname sein, der in der Datenbasis (Hosts-Datei, DNS, ...) generiert ist. Falls Sie die Funktion „Mapped Hostnamen“ verwenden wollen, müssen Sie hier den UTM-Hostnamen angeben (siehe Seite 118). Weitere in der Datenbasis (Hosts-Datei, DNS,...) angegebene Namen, sogenannte Aliasnamen, dürfen Sie hier nicht angeben.

X/W X/W
  – In TRANSPORT-SELECTOR= müssen Sie den T-Selektor des Client angeben. Dieser Name muss mit der Generierung auf der Client-Seite abgestimmt werden.

X/W X/W
  – Bei TSEL-FORMAT= geben Sie den Formatindikator des T-Selektors an. Es wird empfohlen, den Operanden TSEL-FORMAT= immer zu versorgen.

X/W X/W X/W X/W
  – Sofern benötigt, müssen Sie bei SESSION-SELECTOR= und PRESENTATION-SELECTOR= den Session- bzw. Presentation-Selektor des OpenCPIC-Client angeben. Diese Namen müssen mit der Generierung auf der Client-Seite abgestimmt werden.

X/W
  Werden diese Selektoren nicht benötigt, geben Sie jeweils *NONE an.

X/W
  – Bei LISTENER-PORT geben Sie die Portnummer des OpenCPIC-Client an.

X/W
  – Bei T-PROT müssen Sie immer RFC1006 angeben.

5.1.13 Beispiele für die Generierung eines Client/Server-Verbundes

Die folgenden beiden Beispiele zeigen, wie Sie einen UPIC-Client, der auf einem Windows-System läuft, an eine UTM-Anwendung in BS2000 und auf einem Unix-System anschließen.

B **Beispiel 1: UPIC-Client an openUTM im BS2000 anschließen**

B Die UTM-Server-Anwendung befindet sich auf einem Host mit dem Namen BS2HOST1, B das Client-Programm läuft auf einem PC mit dem Rechnernamen PCCLT002 ab. Die Trans- B port-Verbindung soll über TCP/IP aufgebaut werden (Adressformat RFC1006).

B • UTM-Generierung unter BS2000/OSD

```
B   *** BCAM-Anwendungsnamen für die UTM-Server-Anwendung definieren:***
B   BCAMAPPL SERVER, T-PROT=RFC1006
```

```
B   *** Client generieren:***
B   PTERM UPICTTY, PTYPE=UPIC-R, LTERM=UPICLT, BCAMAPPL=SERVER, -
B         PRONAM=PCCLT002
B   LTERM UPICLT
```

```
B   *** TAC für den Client definieren ****
B   TAC TAC1, PROGRAM=SERVICE
```

B Durch die Anweisung LTERM UPICLT erzeugt openUTM beim Anmelden implizit eine B Verbindungsbenutzerkennung namens UPICLT.

B Bei der Kopplung über das RFC1006-Protokoll werden keine BCMAP-Einträge B benötigt.

B • Einträge in die Side Information Datei (upicfile) des openUTM-Client

```
B   * UTM-Anwendung unter BS2000/OSD
B   SDsamplaw SERVER.BS2HOST1 TAC1
B   * oder, falls automatische Konvertierung der Benutzerdaten
B   * zwischen ASCII und EBCDIC gewünscht wird
B   HDsamplaw SERVER.BS2HOST1 TAC1
```

B • Angabe im Client-Programm

```
B   Enable_UTM_UPIC "UPICTTY"
B   Initialize_Conversation "samplaw"
```

Beispiel 2: UPIC-Client an openUTM auf einem Unix-System anschließen

Dieses Beispiel beschreibt die TCP/IP-RFC1006-Kopplung eines UPIC-Client mit einer UTM-Anwendung auf einem Unix-System. Das Beispiel zeigt die Abstimmung der Generierung bei beiden Kommunikationspartnern.

Die UTM-Anwendung läuft auf dem Rechner mit dem Namen `UXHOST01`. Der Client befindet sich auf einem Windows-System, für das in der KDCDEF-Generierung der Name `PCCLT001` generiert wird. Die UTM-Anwendung erhält lokal die Portnummer 1230, der Client die Portnummer 1240.

- Generierung des UTM-Servers am Unix-System

```
BCAMAPPL UTMUPICR,LISTENER-PORT=1230,T-PROT=RFC1006,TSEL-FORMAT=T

PTERM UPICTTY,PTYPE=UPIC-R,LTERM=UPIC,BCAMAPPL=UTMUPICR,PRONAM=PCCLT001, \
            LISTENER-PORT=1240,T-PROT=RFC1006,TSEL-FORMAT=T
LTERM UPIC,USER=UPICUSER
USER  UPICUSER,PERMIT=ADMIN
```

Mit der Anweisung USER UPICUSER... wird eine explizite Verbindungs-Benutzerkennung generiert.

- Einträge in die Side Information Datei des Client-Rechners

```
* Lokale Anwendung
LNUPICTTY UPICTTY PORT=1240

* UTM-Anwendung auf einem Unix-System mit Port 1230, TCP/IP-
Hostname=UXHOST01
SDsampladm UTMUPICR.UXHOST01 KDCHELP PORT=1230
```

- Einträge in der Hosts-Datei des Client-Rechners

In der Datei HOSTS des Windows-Systems muss für die Abbildung des Rechnernamens UXHOST01 auf die IP-Adresse des Unix-Systems folgender Eintrag gemacht werden:

ip-adresse `UXHOST01`

- Angabe im Client-Programm

```
Enable_UTM_UPIC "UPICTTY"
Initialize_Conversation "sampladm"
```

B/X **Beispiel 3: OpenCPIC-Client anschließen**

B/X Ein OpenCPIC-Client läuft auf dem Unix-System mit dem Hostnamen UNIXPRO1. Der
B/X Client schließt sich über RFC1006 an eine UTM-Anwendung auf BS2000/OSD und an eine
B/X UTM-Anwendung auf einem Unix-System an. Dabei soll Folgendes gelten:

B/X – In der UTM-Anwendung auf BS2000/OSD ruft der Client den Transaktionscode
B/X TRAVEL02 und in der UTM-Anwendung auf dem Unix-System den Transaktionscode
B/X STATIST1 auf.

B/X – Es sollen zum BS2000 bis zu 10 parallele und zum Unix-System bis zu 2 parallele logi-
B/X sche Verbindungen möglich sein.

B/X – Der OpenCPIC-Client und die UTM-Anwendung auf BS2000 verwenden lokal Portnum-
B/X mer 102. Die UTM-Anwendung auf dem Unix-System verwendet lokal Portnummer
B/X 12000.

B ● UTM-Generierung auf BS2000/OSD

```
B   UTMD APT = (2, 7, 16, 2)

B   ACCESS-POINT SERVER,
B      T-PROT = RFC1006,
B      P-SEL = *NONE,
B      S-SEL = *NONE,
B      T-SEL = C'UTMSERV1',
B      AEQ = 1

B   OSI-CON CONNECTB,
B      LOCAL-ACCESS-POINT = SERVER,
B      P-SEL = *NONE,
B      S-SEL = *NONE,
B      T-SEL = C'CPICCLT1',
B      N-SEL = C'UNIXPRO1',
B      OSI-LPAP = OSILPAPB

B   OSI-LPAP OSILPAPB,
B      APT = (2, 7, 16, 4),
B      APPLICATION-CONTEXT = UDTSEC,
B      AEQ = 1,
B      ASS-NAMES=CPIC,
B      ASSOCIATIONS=10,
B      CONTWIN=0

B   TAC TRAVEL02 ...
```

B Bei der Kopplung über das RFC1006-Protokoll mit Portnummer 102 werden keine
B BCMAP-Einträge benötigt.

X • UTM-Generierung auf dem Unix-System

```
X   UTMD APT = (2, 7, 16, 3)

X   ACCESS-POINT STATSERV,
X      T-PROT = RFC1006,
X      P-SEL = *NONE,
X      S-SEL = *NONE,
X      T-SEL = C'UTMSERV2',
X      LISTENER-PORT = 12000,
X      T-PROT = RFC1006,
X      T-SEL-FORMAT = T,
X      AEQ = 1

X   OSI-CON CONNECTX,
X      LOCAL-ACCESS-POINT = STATSERV,
X      P-SEL = *NONE,
X      S-SEL = *NONE,
X      T-SEL = C'CPICCLT1',
X      N-SEL = C'UNIXPRO1',
X      LISTENER-PORT = 102,
X      T-PROT = RFC1006,
X      T-SEL-FORMAT = T,
X      OSI-LPAP = OSILPAPX

X   OSI-LPAP OSILPAPX,
X      APT = (2, 7, 16, 4),
X      APPLICATION-CONTEXT = UDTSEC,
X      AEQ = 1,
X      ASS-NAMES=CPIC,
X      ASSOCIATIONS=2,
X      CONTWIN=0

X   TAC STATIST1 ...
```

B/X • OpenCPIC-Generierung

```
B/X *** Eintrag für die lokale Anwendung
B/X LOCAPPL OPENCPIC,
B/X    APT = (2, 7, 16, 4),
B/X    AEQ = 1

B   *** Verbindung zur UTM-Anwendung unter BS2000/OSD
B   PARTAPPL UTMSBS20,
B      APT = (2, 7, 16, 2),
B      APPLICATION-CONTEXT = utm-secu,
B      AEQ = 1,
B      ASSOCIATIONS = 10,
B      CONTWIN = (10,10),
B      CONNECT = 10
```

B
```
*** TAC in der UT-MAnwendung unter BS2000/OSD
SYMDEST TRAVEL,
   PARTNER-APPL = UTMSBS20,
   PARTNR-APRO = TRAVEL02
```

X
```
*** Verbindung zur UTM-Anwendung auf einem Unix-System
PARTAPPL UTMSUNIX,
   APT = (2, 7, 16, 3),
   APPLICATION-CONTEXT = utm-secu,
   AEQ = 1,
   ASSOCIATIONS = 2,
   CONTWIN = (2,2),
   CONNECT = 2
```

X
```
*** TAC in der UTM-Anwendung auf einem Unix-System
SYMDEST STATIST,
   PARTNER-APPL = UTMSUNIX,
   PARTNR-APRO = STATIST1
```

B/X

- TNS-Einträge im OpenCPIC Client-Rechner (tnsxfrm-Format)

B/X
```
OPENCPIC\
   PSEL V''
   SSEL V''
   TSEL RFC1006 T'CPICCLT1'
```

B
```
UTMSBS20\
   PSEL V''
   SSEL V''
   TA RFC1006 ip-adresse-bs2 PORT 102 T'UTMSERV1'
```

X
```
UTMSUNIX\
   PSEL V''
   SSEL V''
   TA RFC1006 ip-adresse-unix PORT 12000 T'UTMSERV2'
```

5.2 Drucker generieren (auf BS2000/OSD und Unix-Systemen)

W W

In UTM-Anwendungen unter Windows-Systemen können keine Drucker generiert werden.

B/X B/X B/X B/X B/X Drucker, die von einer UTM-Anwendung genutzt werden sollen, werden über LTERM-Partner angeschlossen, die mit den logischen Eigenschaften für Drucker konfiguriert werden. LTERM-Partner für Drucker definieren Sie in der LTERM-Anweisung. Drucker können **nicht** über LTERM-Pools angeschlossen werden. Physische Drucker werden mit der PTERM-Anweisung definiert, hier findet auch die Zuordnung zum LTERM-Partner statt.

B/X B/X B/X Der Verbindungsaufbau durch openUTM kann entweder mit PTERM...,CONNECT=YES oder mit LTERM...,PLEV= festgelegt werden. Die Verbindung kann auch administrativ aufgebaut werden. Sehen Sie dazu das openUTM-Handbuch „Anwendungen administrieren“.

B/X B/X B/X

LTERM-Anweisung auf Seite 354
Die wichtigsten Eigenschaften für LTERM-Partner, über die sich Drucker an eine Anwendung anschließen können, werden mit den folgenden Operanden definiert:

B/X
- ltermname

  B/X B/X Name des LTERM-Partners, über den der Drucker an die UTM-Anwendung angeschlossen wird.

B/X
- CTERM=

  B/X B/X B/X Das Druckersteuer-LTERM festlegen, damit der Benutzer Drucker, Druckaufträge und Druckaufträge in der Message Queue des LTERM-Partners administrieren kann.

B/X
- PLEV=

  B/X B/X Anzahl der Drucker-Nachrichten, bei der openUTM versucht, die Verbindung zum Drucker aufzubauen, der diesem LTERM-Partner zugeordnet ist.

  B/X – Bei PLEV=1 wird bei jedem Druckauftrag eine Verbindung aufgebaut.

  B/X B/X – Bei PLEV=n wird beim n-ten Druckauftrag die Verbindung aufgebaut (n=1 bis 32767).

  B/X B/X B/X – Bei PLEV=0 wird der Verbindungsaufbau nicht durch anstehende Druckaufträge initiiert, sondern vom Administrator explizit mit dem Kommando KDCLTERM...ACT=CON bzw. mit dem Kommando KDCPTERM veranlasst.

  B/X B/X Nachdem alle anstehenden Druckaufträge erledigt sind, wird die Verbindung zum Drucker wieder abgebaut.

B/X PLEV= erleichtert dem Benutzer, Drucker von verschiedenen UTM-Anwendungen aus zu benutzen (Drucker Sharing). Dazu wird die Verbindung zwischen einer UTM-Anwendung und dem Drucker nur für die Dauer der Übermittlung des Druckauftrags aufrechterhalten, um anderen Anwendungen die Möglichkeit zum Verbindungsaufbau zu geben. Wird die UTM-Anwendung beendet und stehen noch unbearbeitete Druck-Nachrichten für einen Drucker in der Message Queue des LTERM-Partners, bleiben diese Druck-Nachrichten bis zum nächsten Anwendungsstart erhalten. Der Administrator kann, bevor er die Anwendung beendet, die Bearbeitung der ausstehenden Druck-Nachrichten mit Hilfe des Kommandos KDCAPPL SPOOLOUT=ON veranlassen.

B/X
- QAMSG=

B/X Nachrichten an Drucker können in der Message Queue des LTERM-Partners zwischengespeichert werden, auch wenn der Drucker nicht mit der Anwendung verbunden ist.

B/X
- USAGE=O

B/X Mit USAGE=O wird ein Drucker als Kommunikationspartner festgelegt, der sich über den LTERM-Partner anschließen kann. Es können nur Nachrichten von der Anwendung zum Drucker gesendet werden.

B/X Jeder Drucker muss in der Konfiguration beschrieben sein, d.h. für jeden Drucker wird eine PTERM-Anweisung mit den physischen Eigenschaften des Druckers geschrieben und die Zuordnung zu einem LTERM-Partner vorgenommen. Die Zuordnung zwischen Drucker und LTERM-Partner ist statisch, d.h. die Zuordnung bleibt solange erhalten, bis sie per Administration aufgehoben wird. Sie können einem LTERM-Partner im laufenden Betrieb einen anderen Drucker (gleichen Typs) zuordnen, beispielsweise bei einer Druckerstörung.

B/X

PTERM-Anweisung auf Seite 440

B/X Die wichtigsten Eigenschaften für Drucker werden mit den folgenden Operanden festgelegt:

B/X
- ptermname

B/X Name des Druckers

B *BS2000/OSD:*
B Es ist der BCAM-Name oder der RSO-Name des Druckers anzugeben.

X *Unix-Systeme:*
X Es ist der Name einer Druckergruppe des Spoolsystems anzugeben. Eine von
X UTM-Anwendungen verwendete Druckergruppe sollte nur aus **einem** Drucker be-
X stehen. Nur so ist immer gewährleistet, dass bei der Ausgabe von Nachrichten, die
X sich aus Teil-Nachrichten zusammensetzen, alle Teile einer Nachricht auf demsel-
X ben Drucker ausgegeben werden (siehe hierzu auch Druckerbündel).

B/X
- CID=

B/X
B/X
B/X
Dem Drucker wird eine Drucker-ID *printer_id* zugeordnet, über die der Drucker von einem Druckersteuer-LTERM identifiziert werden kann. Das Druckersteuer-LTERM „hängt“ am LTERM-Partner, dem der Drucker zugeordnet wird.

B/X
- CONNECT=

B/X
B/X
B/X
gibt an, ob openUTM beim Anwendungsstart automatisch eine Verbindung zum Drucker aufbaut. Der Drucker ist dann durch die Anwendung bis zum nächsten Verbindungsabbau explizit belegt, auch wenn keine Druckaufträge anstehen.

B/X
- LTERM=

B/X
B/X
Name des LTERM-Partners, der dem Drucker *ptermname* zugeordnet wird und über den der Drucker an die UTM-Anwendung angeschlossen wird.

B/X
- PTYPE=

B
B
BS2000/OSD
Druckertyp oder *RSO.

X
X
X
X
X
X
Unix-Systeme:
Druckertyp.
Für die Ausgabe der Daten ruft der Druckerprozess (*utmprint*) das Skript *utmlp* auf. Beim Aufruf werden zusätzlich zu den auszudruckenden Daten Parameter an *utmlp* übergeben. *utmlp* übergibt dann standardmäßig die Daten an das Kommando lp (siehe PTYPE=PRINTER auf Seite 456).

B
- USAGE=O (nur BS2000/OSD)

B
Der Kommunikationspartner ist ein Drucker.

B/X B/X Maximal- und Grenzwerte, die anwendungsweit für Drucker gelten sollen, legen Sie mit der MAX-Anweisung fest.

B/X B/X B/X **MAX-Anweisung** auf Seite 368
Standard- und Maximalwerte, die anwendungsweit für Drucker relevant sind, werden mit den folgenden Operanden festgelegt:

B/X
- CONRTIME=

  B/X B/X Zeit in Minuten, nach der openUTM zyklisch versucht, eine logische Verbindung wieder aufzubauen. openUTM versucht dies für:

  B/X B/X B/X B/X B/X
  – Drucker, zu denen openUTM eine Verbindung aufbaut, sobald die Anzahl der Druckaufträge für diesen Drucker den generierten Schwellwert erreicht (LTERM...,PLEV>0). Ist dieser Schwellwert erreicht, versucht openUTM die Verbindung zum Drucker auch dann aufzubauen, wenn diese vorher explizit durch die Administration abgebaut wurde.

  B/X B/X B/X
  – Drucker, zu denen openUTM automatisch eine Verbindung aufbaut (PTERM...,CONNECT=YES) wenn die Verbindung nicht durch die Administration abgebaut wurde.

  B/X B/X B/X Kommt beim Start der Anwendung keine Verbindung zustande oder wird die Verbindung im laufenden Betrieb unterbrochen, so versucht openUTM im Abstand von CONRTIME= die Verbindung wiederaufzubauen.

B/X
- PGPOOL=

  B/X B/X B/X Beim Operanden PGPOOL ist ein ausreichend großer Wert anzugeben, damit bei hohem Druck-Aufkommen der Pagepool alle Druck-Nachrichten aufnehmen kann und nicht überläuft.

B/X
- TRMSGLTH=

  B/X B/X Maximale Länge der physischen Nachrichten, die von der UTM-Anwendung und Druckern ausgetauscht werden können.

B
- LOGACKWAIT=

  B B Zeit in Sekunden, die openUTM maximal auf eine Quittung von Ausgabegeräten warten soll. Diese Quittung ist

  B B B
  – bei einem Drucker die logische Abdruckquittung vom Drucker,
  – bei einem RSO-Drucker die Quittung von RSO,
  – bei einem FPUT-Aufruf an eine andere Anwendung die Transportquittung.

5.2.1 RSO-Drucker generieren (BS2000/OSD)

B Über die OLTP-Schnittstelle von RSO (Remote Spool Output) erhält openUTM Zugang zu allen Druckern, die RSO unterstützt, d.h. auch zu Druckern, die über LAN oder PC angeschlossen sind. Zu diesen Druckern baut openUTM keine Transportverbindung auf, sondern bedient sie über die OLTP-Schnittstelle, d.h. openUTM reserviert den Drucker bei RSO und übergibt den Druckauftrag an RSO.

5.2.1.1 Einträge für die KDCDEF-Generierung

B Um von openUTM aus auf einem RSO-Drucker auszudrucken, wird bei der Generierung der gewünschte Drucker in der PTERM-Anweisung unter seinem RSO-Namen als RSO-Drucker definiert. Auf RSO-Seite muss der Drucker definiert und aktiviert sein. Dieser Abschnitt führt nur die RSO-spezifischen Anweisungen und Operanden auf, die anderen Drucker-spezifischen Parameter sind auf Seite 165f beschrieben.

B

PTERM-Anweisung auf Seite 440
RSO-Drucker, die openUTM über die OLTP-Schnittstelle bedient, werden mit den folgenden Operanden definiert:

B
- ptermname

  B Für einen RSO-Drucker muss hier der Name des Druckers angegeben werden, wie er in RSO definiert wurde (logischer RSO-Device-Name).

B
- PTYPE=

  B Als Druckertyp wird PTYPE=*RSO angegeben. Für RSO-Drucker wird kein bestimmter Druckertyp angegeben. openUTM erhält den Druckertyp entsprechend der RSO-Device-Information beim RSO-Aufruf.

B
- PRONAM=

  B Für einen RSO-Drucker muss als Rechnername *RSO angegeben werden.

B

MAX-Anweisung auf Seite 368
Grenzwerte, die in der Anwendung für Drucker relevant sind, werden mit den folgenden Operanden festgelegt:

B
- TRMSGLTH=

  B Maximale Länge der physischen Nachrichten, die zwischen der UTM-Anwendung und Druckern ausgetauscht werden können.
  Wenn RSO-Drucker definiert werden, muss REMOTE-BUFFER-SIZE in der SPOOL-Parameterdatei ≥ TRMSGLTH sein.

5.2.1.2 Einträge bei RSO und SPOOL

Voraussetzung für die Nutzung der OLTP-Schnittstelle von RSO durch openUTM ist die Installation von RSO ab V3.0A und der von RSO vorausgesetzten Software-Produkte. Das Subsystem RSO muss aktiv sein. Lesen Sie zu Detailfragen das Handbuch zu RSO.

Device-Definition

Mit dem Dienstprogramm SPSERVE öffnen Sie die SPOOL-Parameterdatei für die Druckerdefinition. Der Systemverwalter muss den Drucker in RSO für UTM-Druckaufträge konfigurieren:

```
ADD-SPOOL-DEVICE...ADMINISTRATOR=*ADMINISTRATOR(...),
                   PROCESSING-CONTROL=*PAR(
                                      DISCONNETION=*YES
                                      RESET={*YES | *NO }
                                      CONTROLLER-START=AT-PRINTER-START)
```

- Bei der Definition eines Druckers können mit dem Parameter ADMINISTRATOR= *ADMINISTRATOR(...) bis zu 8 RSO-Geräteverwalter eingetragen werden. Ein RSO-Geräteverwalter darf ein Device mit MODIFY-SPOOL-DEVICE modifizieren oder mit START-PRINTER-OUTPUT starten.
- Es wird empfohlen, mit dem Parameter DISCONNECT=*YES zu arbeiten, da der Drucker bei SOCKETS die Verbindung abbaut, wenn die am Drucker eingestellte Wartezeit abgelaufen ist.
- Der Parameter CONTROLLER-START muss auf AT-PRINTER-START gesetzt werden.
- Bei RESET=*YES werden die Einstellungen des Druckermenüs verwendet. Dies geschieht auch unabhängig vom Device-Eintrag, wenn openUTM mit Formaten arbeitet. Werden „logische“ Formate verwendet, siehe auch Hinweis am Ende des Abschnitts, dann verhält sich openUTM wie im Fall ohne Formate.
  - Wird von openUTM bei einem FPUT im Feld KCMF ein Formatname übergeben, dann schickt FHS standardmäßig ein RESET=*YES vor der Nachricht an den Drucker, damit die Menüeinstellung des Druckers vor dem Ausdruck wirksam wird. Am Druckermenü können beispielsweise verschiedene Fonts oder CPI-Werte eingestellt sein. RSO bearbeitet in diesem Fall eine Nachricht mit Formatnamen wie mit der Einstellung CONTROL=TRANSPARENT.
  - Wird von openUTM bei einem FPUT im Feld KCMF kein Formatname übergeben, dann schickt RSO nur ein RESET vor der Nachricht an den Drucker, wenn in der Device-Definition RESET=*YES eingetragen ist. Wenn kein RESET an den Drucker geschickt wurde, gelten dabei die gerade am Drucker eingestellten Werte des Druckermenüs, die durch einen vorherigen Druckauftrag verändert worden sein können. RSO behandelt eine Drucker-Nachricht ohne Formatnamen wie mit der Einstellung CONTROL=PHYSICAL.

B Die Kommandos ADD-SPOOL-FORM für Formulareinträge und ADD-SPOOL-CHARACTER für Zeichensätze sind für UTM-Druckaufträge wirkungslos. Wenn unter dem gleichen logischen RSO-Device-Namen sowohl UTM-RSO-Druckaufträge als auch RSO-SPOOLOUT abgewickelt werden, so sind Formulare und Zeichensätze nur für letzteren relevant. RSO ergänzt für UTM-Druckaufträge als einziges Druckersteuerzeichen die RESET-Zeichenfolge.

B Ein UTM-Druckauftrag an einen RSO-Drucker wird nicht in die SPOOL-Queue eingereiht.

B Ab FHS V08.2C können Formate auf **alle** RSO-Drucker ausgegeben werden, wenn die Formate entsprechend erstellt wurden (ab IFG V8.1B), siehe auch Handbuch „FHS - Formatierungssystem für openUTM, TIAM, DCAM“. Dabei können Formate so erzeugt werden, dass sie als „logische“, nicht-Geräte-spezifische Formate an RSO geschickt werden.

B **Beispiel eines Device-Eintrags**

B Ausgabe eines Device-Eintrags unter dem der Drucker für RSO definiert ist.

```
B  /show-spool-dev PGTP0041,inf=*all
B  DEVICE-NAME          : PGTP0041
B  DEVICE-TYPE          : 9021RP
B  ACCESS-DATE          : 2006-11-27
B  ------------------------------- DEVICE-ACCESS -----------------------------
B  DEVICE-ACCESS        : *TCP-ACCESS
B  ACCESS-TYPE          : *TACLAN
B  PROCESSOR-NAME       : *NONE
B  STATION-NAME         : *NONE
B  MNEMONIC-NAME        : *NONE
B  PROGRAM-NAME         : *NONE
B  INTERNET-ADDRESS     : PGTP0041
B  PORT-NAME            : 9100
B  LPD-PRINTER-NAME     : *NONE
B  FROM-PORT-NUMBER     : 0
B  TO-PORT-NUMBER       : 0
B  ------------------------------ TWIN-DEVICE-DEF ----------------------------
B  SLAVE-MNEMONIC-NAME : *NONE
B  ESD-SIZE             : 0
B  ---------------------------- DEVICE-INFORMATION ---------------------------
B  FORMS-OVERLAY-BUFFER: 32767
B  CHARACTER-SET-NUMBER: 64
B  ROTATION             : NO
B  DUPLEX-PROCESSING    : NO
B  FORMS-OVERLAY        : NO
B  RASTER-PATTERN-MEM   : *NONE
B  TRANSMISSION         : IGN
B  FONT-TYPE            : IGN
B  FACE-PROCESSING      : NO
```

```
B  MAXIMUM-INPUT-TRAY  : 1
B  MONJV               : NO
B  NOTIFICATION        : NO
B  ENCRYPTION          : NO
B  UNICODE             : NO
B  SUPP-FORMAT-NAME    :
B     TEXT
B     PLAIN-TEXT
B  ------------------------------- ADMINISTRATOR -----------------------------
B  USER-IDENTIFICATION : *NONE
B
B  IDENTIFICATION      : OEC MW 135
B  TERMINAL            : PROCESSOR-NAME      :
B                        STATION-NAME        :
B  ------------------------------ SPOOLOUT-CONTROL ----------------------------
B  SHIFT               : 0
B  LINE-FEED-COMPRESS  : YES
B  BLANK-COMPRESSION   : YES
B  START-FORM-FEED     : YES
B  FORM-FEED           : *SINGLE-SHEET
B                        DEFAULT-TRAY-NUMBER : 1
B                        OUTPUT-TRAY-NUMBER  : 0
B  SKIP-TO-CHANNEL     : OPTIM
B  SKIP-TO-NEXT-PAGE   : BY-FORM-FEED
B  ESCAPE-VALUE        : NONE
B  ---------------------------- PROCESSING-CONTROL ----------------------------
B  CONTROLLER-RESERVED : NO
B  FORM-NAME           : STD
B
B  DISCONNECTION       : YES
B  BUFFER-SIZE         : 1024
B  RESET               : YES
B  REPEAT-MESSAGE      : TYPE               : SYS
B                        LIMIT              : NO
B                        RETRY-TIME         : GLB
B  RESTART-ACTION      : LIMIT              : NO
B                        RETRY-TIME         : GLB
B  SYNCHRONIZATION     : PRINTER
B  TIMEOUT-MAX         : 2
B  PAGE-EJECT-TIMEOUT  : NO
B  BAND-IDENTIFICATION : *NONE
B  LOAD                : NO
B  MODULO2             : NO
B  RECOVERY-RULES      : *SYSTEM
B  POLLING             : NO
B  PRINTER-PARAM-FILE  : *SYSTEM
B  RESOURCE-FILE-PREFIX: *SYSTEM
B  CONTROLLER-START    : AT-PRINTER-START
```

```
B   ------------------------------ CHARACTER-SET-POS ------------------------------
B   POSITION-1          : N-U
B   POSITION-2          : N-U
B   POSITION-3          : N-U
B   POSITION-4          : N-U
B   POSITION-5          : N-U
B   POSITION-6          : N-U
B   POSITION-7          : N-U
B   POSITION-8          : N-U
B   POSITION-9          : N-U
B   POSITION-10         : N-U
B   POSITION-11         : N-U
B   POSITION-12         : N-U
B   POSITION-13         : N-U
B   POSITION-14         : N-U
B   POSITION-15         : N-U
B   POSITION-16         : N-U
B   ------------------------------- MISCELLANEOUS -------------------------------
B   REDIRECTION-DEVICE  : *NONE
B   LANGUAGE-EXT-TYPE   : *SYSTEM
B   LINE-SIZE           : 150
B   CHARACTER-IMAGE     : *NONE
```

B **RSO-Puffergröße festlegen**

B Um Nachrichten beliebiger Länge ausdrucken zu können, muss der RSO-Puffer größer gleich der maximalen Nachrichtenlänge bei openUTM sein (MAX ...,TRMSGLTH=). Da der Maximalwert für die UTM-Puffergröße 32 KB ist, muss die RSO-Puffergröße in einer Session, in der openUTM läuft, diesem Wert angepasst werden:

```
B   /MODIFY-SPOOL-PARAMETER...SPOOLOUT-OPTIONS=*PAR(REMOTE-BUFFER-SIZE=32)
```

B Es kann auch ein kleinerer Wert gewählt werden, wenn bei openUTM kleinere Werte generiert sind. Eine Änderung wird in der nächsten SPOOL-Session wirksam.

B **VTSU-Codes**

B Wenn UTM-Nachrichten, die VTSU-Codes enthalten, auf direkt über BCAM angeschlossenen Druckern ausgedruckt werden, ruft openUTM das Programm VTSU auf, um VTSU-Codes in druckerabhängige Escape-Sequenzen umzusetzen. Wenn UTM-Nachrichten auf RSO-Druckern ausgedruckt werden, wird die Umsetzung der VTSU-Codes von RSO vorgenommen. Dabei wurde eine weitgehende Anpassung an die bekannten VTSU-Steuerzeichen angestrebt. Weitere Informationen finden Sie im RSO-Handbuch.

5.2.1.3 Drucker für openUTM aktivieren

Jeder Drucker, der in einer RSO-Session verwendet wird, muss mit START-PRINTER-OUTPUT gestartet werden. Eine START-PRINTER-OUTPUT-Anweisung kann in einer ENTER-Datei stehen, die beim Start des RSO-Subsystems automatisch abgearbeitet wird, oder der Systemverwalter bzw. RSO-Geräteverwalter startet den Drucker nach dem Start des Subsystems explizit mit dieser Anweisung.

Wenn von openUTM aus auf einem RSO-Drucker gedruckt werden soll, muss der Drucker für openUTM freigegeben werden:

```
/START-PRINTER-OUTPUT DEVICE-NAME=*RSO(NAME=devicename,
                                       ALLOWED-ACCESSES='UTM')
                                 oder  ALLOWED-ACCESSES=('RSO','UTM'))
```

5.2.1.4 Druckerinformationen abfragen

Druckerinformationen bei openUTM

Mit dem Administrationskommando KDCINF kann in openUTM der aktuelle Druckerstatus abgefragt werden. Im Bedarfsfall kann mit den Administrationskommandos KDCPTERM und KDCLTERM oder über die Programmschnittstelle zur Administration die Verbindung zum Drucker auf- oder abgebaut oder gesperrt werden. Sehen Sie dazu auch openUTM-Handbuch „Anwendungen administrieren“.

Druckerinformationen bei RSO

Benutzer und RSO-Geräteverwalter können sich im BS2000/OSD Informationen über den Druckerstatus der RSO-Drucker mit dem folgenden Kommando ausgeben lassen:

```
/SHOW-SYSTEM-STATUS INFORMATION=*REMOTE(DEVICE=NAME)
```

5.2.1.5 Drucker freigeben im Fehlerfall

B Wenn im Fehlerfall die automatische Druckauftragswiederholung von openUTM und RSO
B nicht erfolgreich war, kann der RSO-Geräteverwalter den Drucker mit dem Kommando

```
B  /STOP-PRINTER-OUTPUT DEVICE-NAME=RSO-PRINTER(NAME=rso-printer,STOP=IMMEDIATE)
```

B kurzfristig freischalten, um den Drucker anschließend mit dem Kommando
B START-PRINTER-OUTPUT wieder für openUTM zu reservieren.

B Zu Diagnosezwecken kann unter $TSOS der Parameter TRACE=YES angegeben werden.
B Es wird dann ein Trace erzeugt, der abgelegt wird unter:
B $SYSSPOOL.SYSTRC.RSO.*devicename.datum.uhrzeit*

```
B  /START-PRINTER-OUTPUT DEVICE-NAME=*RSO(NAME=devicename,
B                                        TRACE=YES,
B                                        ALLOWED-ACCESSES=('RSO','UTM'))
```

B Sehen Sie dazu auch die RSO-Handbücher.

5.2.2 Druckerbündel generieren

B/X Ein Druckerbündel besteht aus mehreren Druckern (= Druckergruppen in Unix-Systemen),
B/X die **einem** LTERM-Partner zugeordnet sind. Für jeden Drucker des Bündels schreibt man
B/X eine PTERM-Anweisung mit dem gleichen *ltermnamen* bei PTERM...,LTERM=.
B openUTM verteilt die Druckausgaben möglichst gleichmäßig auf die Drucker des Bündels.
B Nachrichten, die aus Teil-Nachrichten zusammengesetzt sind, gibt openUTM immer voll-
B ständig auf **einen** Drucker bzw. unter Unix-Systemen auf **eine** Druckergruppe des Bündels
B aus.

5.2.3 Bypass-Betrieb (BS2000/OSD)

B Bei einem lokal angeschlossenen Drucker spricht man statt von Spool-Betrieb auch von
B Bypass-Betrieb. Der Bypass-Betrieb ist möglich, wenn das Terminal unabhängig von der
B Druckausgabe einen Dialog führen kann. Bypass-Betrieb ist nur zu realisieren für die Ter-
B minaltypen 975x und 9763 oder eine entsprechende Emulation (siehe Handbuch „MT9750,
B 9750-Emulation unter Windows)“.

5.2.4 Druckersteuer-LTERMs generieren

B/X Bei der Generierung können Sie Druckersteuer-LTERMs definieren, damit Anwender die von ihnen standardmäßig genutzten Drucker und Druckauftrags-Queues auch ohne Administrationsberechtigung selbst administrieren können, beispielsweise den aktuellen Druckauftrag löschen.

B Jedem Drucker wird ein LTERM-Partner zugeordnet, der für ein Ausgabemedium konfiguriert ist (LTERM...,USAGE=O). Alle Ausgabe-Aufträge für diesen Drucker sendet openUTM in die Message Queue des zugeordneten LTERM-Partners, die damit zur Druckauftrags-Queue wird. Einem LTERM-Partner können Sie auch mehrere Drucker zuordnen (Druckerbündel). Dann arbeiten alle Drucker mit dieser Druckauftrags-Queue.

B/X Ein Druckersteuer-LTERM ist ein LTERM-Partner, der als Dialog-Partner konfiguriert ist (LTERM...,USAGE=D). Über diesen LTERM-Partner kann sich ein Client oder ein Terminal-Benutzer an die Anwendung anschließen, um Drucker und die zugehörigen Druckauftrags-Queues zu administrieren.

B/X Sie ordnen die Drucker über die LTERM-Partner dem jeweiligen Druckersteuer-LTERM zu, d.h. Sie geben für die LTERM-Partner mit LTERM...,CTERM=*druckersteuer-ltermname* das Druckersteuer-LTERM an, dem die Drucker zugeordnet werden.

B/X Damit die Druckersteuer-LTERMs die ihnen zugeordneten Drucker identifizieren können, vergeben Sie jedem Drucker in der PTERM-Anweisung eine Control Identification (CID). Diese CID muss im Bereich eines Druckersteuer-LTERMs eindeutig sein, da das Druckersteuer-LTERM die Drucker über die Drucker-Id anspricht. Auf die Eindeutigkeit der Drucker-Id müssen Sie insbesondere bei Druckerbündeln achten. Sie müssen für jeden Drucker des Bündels eine eigene Drucker-Id definieren, die zu keinem anderen Drucker des Druckersteuer-LTERMs gehört.

B/X Um den Zugang zum Druckersteuer-LTERM auf einen bestimmten Personenkreis zu beschränken, können Sie dem Druckersteuer-LTERM wie jedem LTERM-Partner einen Lockcode zuordnen.

B/X Für die einem Druckersteuer-LTERM zugeordneten Drucker gibt es ein Quittungsverfahren, welches für jeden einzelnen Drucker bei Bedarf ein- oder ausgeschaltet werden kann. Alle einer Druckersteuerung zugeordneten Drucker laufen bei ihrem ersten Anwendungsstart nach einer Neugenerierung stets im Automatikmodus.

B/X Weitere Informationen zur Druckeradministration sind dem openUTM-Handbuch „Anwendungen administrieren“ zu entnehmen.

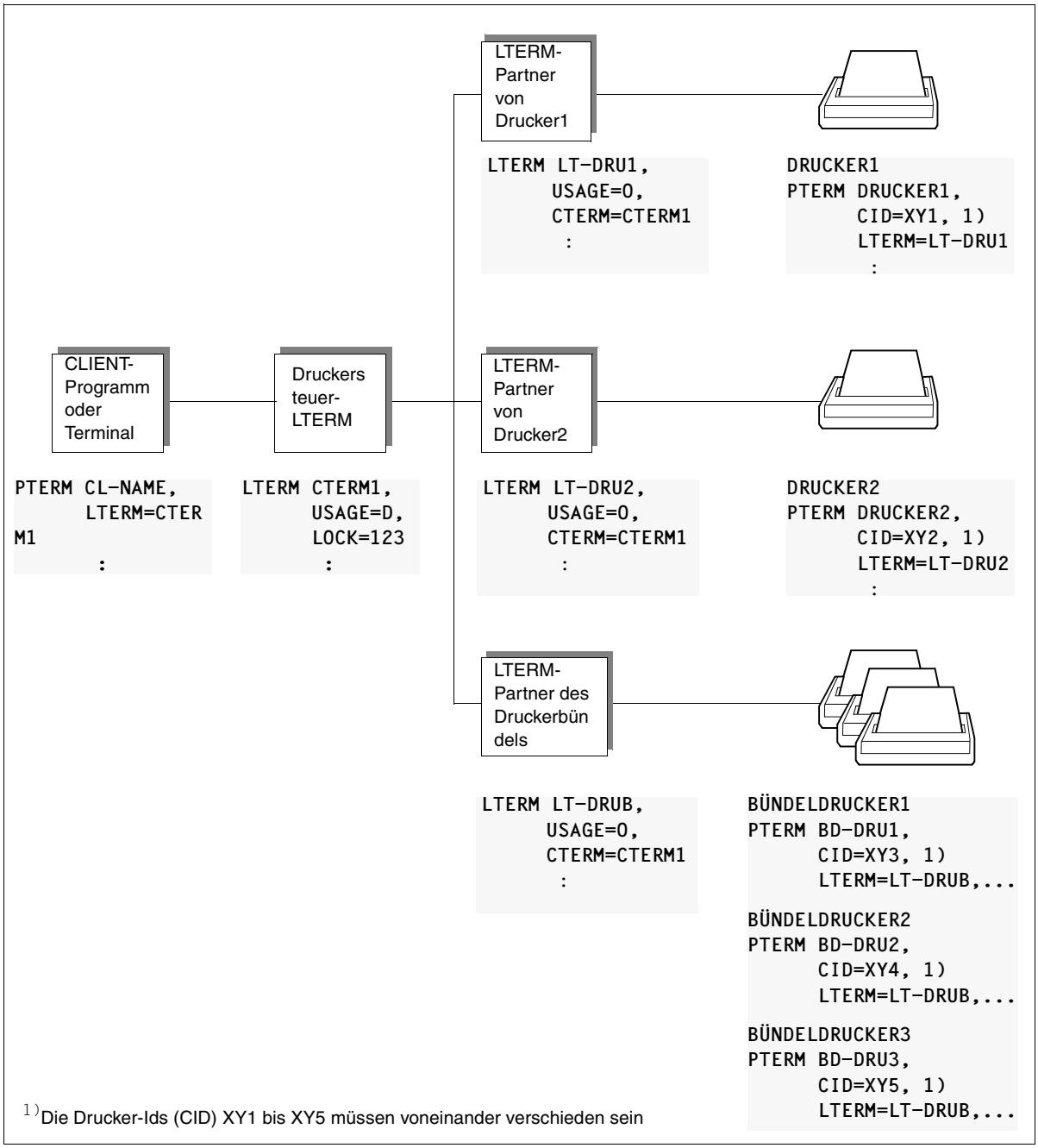


Bild 14: Konfigurierung eines Druckersteuer-LTERMs und der zugehörigen Drucker

5.3 Service-gesteuerte Queues generieren

openUTM bietet Service-gesteuerte Queues an, d.h. Message Queues, deren Verarbeitung von den Teilprogrammen der Anwendung kontrolliert wird. Ein Teilprogramm eines Dialog- oder Asynchron-Vorgangs muss die Nachricht einer Service-gesteuerten Queue von sich aus mit dem KDCS-Aufruf DGET lesen. Ein Vorgang kann auch auf das Eintreffen von Nachrichten warten.

Da die Nachrichten im Pagepool gespeichert werden, müssen Sie den Pagepool ausreichend groß konfigurieren.

openUTM stellt drei verschiedene Service-gesteuerte Queue-Typen zur Verfügung:

– USER-Queues (Benutzer-spezifisch)
– TAC-Queues (über TAC-Anweisungen definiert)
– Temporäre Queues (mit QCRE-Aufrufen angelegt und mit QREL-Aufrufen gelöscht)

Eine allgemeine Einführung in Service-gesteuerte Queues und deren Anwendungsszenarien finden Sie im openUTM-Handbuch „Konzepte und Funktionen“.

Die Realisierung der Anwendungsszenarien ist im openUTM-Handbuch „Anwendungen programmieren mit KDCS“ beschrieben. Dort finden Sie auch Informationen zum Verarbeiten von Service-gesteuerten Queues (Lesen, Schreiben und Löschen).

5.3.1 USER-Queues

Jedem Benutzer einer UTM-Anwendung steht automatisch eine permanente Message Queue zur Verfügung, die mit der Benutzerkennung angesprochen wird.

Für USER-Queues ist mit Q-READ-ACL bzw. Q-WRITE-ACL (USER-Anweisung) ein Zugriffsschutz gegen Lesen bzw. Schreiben generierbar.

USER-Anweisung auf Seite 538
Folgende Operanden stehen für USER-Queues zur Verfügung:

- QLEV=

  Mit QLEV kann eine zu starke Belastung des Pagepools durch Nachrichten für diesen USER verhindert werden.
  QLEV gibt an, wieviele asynchrone Nachrichten in der USER-Queue maximal zwischengespeichert werden können (Voreinstellung: 32767, d.h. keine Beschränkung). Wird der angegebene Wert überschritten, wird das weitere Vorgehen vom Wert des Parameters QMODE bestimmt.

- QMODE=

  Bestimmt das Verhalten von openUTM für den Fall, dass bereits die maximal erlaubte Anzahl von Nachrichten in der USER-Queue gespeichert und somit der Queue-Level (Operand QLEV=) erreicht ist. Mit dem Wert STD werden in diesem Fall alle neuen DPUT-Aufrufe abgewiesen, mit WRAP-AROUND wird die jeweils älteste Nachricht von einer neuen überschrieben.

- Q-READ-ACL=

  Legt Lese- und Lösch-Rechte in der USER-Queue für fremde Benutzer fest. Geben Sie Q-READ-ACL= nicht an, dann hat jeder Benutzer Lese- und Lösch-Rechte in der Queue.

- Q-WRITE-ACL=

  Legt Schreibrechte in der USER-Queue für fremde Benutzer fest.
  Geben Sie Q-WRITE-ACL= nicht an, dann hat jeder Benutzer Schreibrechte in der Queue.

5.3.2 TAC-Queues

Durch Generierung von Transaktionscodes mit TYPE=Q (Queue) entstehen permanente Message Queues mit festem Namen.

Die TAC-Queue mit dem festen Namen KDCDLETQ wird als Dead Letter Queue bezeichnet. openUTM stellt diese TAC-Queue zur Verfügung, um Asynchron-Nachrichten an Transaktionscodes oder TAC-Queues zu sichern, die nicht verarbeitet werden konnten (siehe Seite 495).

TAC-Queues können dynamisch gegen Lesen und/oder Schreiben gesperrt werden (siehe Seite 221).

TAC-Anweisung auf Seite 488
Folgende Operanden sind für die Generierung einer über TAC-Anweisungen definierten Queue von Bedeutung:

- *tacname*

  Name des TACs.

- TYPE=Q

  Für TAC-Queues muss TYPE=Q angegeben werden. Es wird eine Message Queue generiert. In eine solche Queue kann mit einem FPUT- oder DPUT-Aufruf eine Nachricht geschrieben und aus der Queue kann mit einem DGET-Aufruf gelesen werden.

- ADMIN=

  Gibt an, ob für Zugriffe auf diese Queue die Administrationsberechtigung erforderlich ist.

- DEAD-LETTER-Q=

  Gibt an, ob Asynchron-Nachrichten dieser Message Queue nach einer nicht ordnungsgemäßen Verarbeitung in der Dead Letter Queue gesichert werden sollen, wenn die maximale Anzahl erneuter Zustellungsversuche (Anweisung MAX, Parameter REDELIVERY) erreicht ist.

- QLEV=

  Mit QLEV kann eine zu starke Belastung des Pagepools durch Aufträge für diese TAC-Queue verhindert werden.
  QLEV gibt die maximale Anzahl der asynchronen Nachrichten an, die in der Message Queue dieses Transaktionscodes stehen können.

- QMODE=

  Bestimmt das Verhalten von openUTM für den Fall, dass bereits die maximal erlaubte Anzahl von Nachrichten in einer Queue gespeichert und somit der Queue-Level erreicht ist.

- Q-READ-ACL=

  Das Keyset definiert die Rechte, die zum Lesen und Löschen von Nachrichten dieser Queue berechtigen.

- Q-WRITE-ACL=

  Das Keyset definiert die Rechte, die zum Schreiben von Nachrichten in diese Queue berechtigen.

- STATUS=

  Legt fest, ob die Message Queue beim Anwendungsstart gesperrt oder freigegeben ist.

5.3.3 Temporäre Queues

Temporäre Queues werden dynamisch durch Programmaufrufe erzeugt und gelöscht. Dabei kann der Queue-Name wahlweise selbst festgelegt oder von openUTM vergeben werden.

Die maximale Anzahl Temporärer Queues wird bei der Generierung mit der QUEUE-Anweisung festgelegt.

QUEUE-Anweisung auf Seite 461
Mit folgenden Operanden werden Eigenschaften von Temporären Queues definiert:

- NUMBER=

  Legt die maximale Anzahl Temporärer Queues fest, die während eines Anwendungslaufs zu einer Zeit existieren können (1 ≤ NUMBER ≤ 500.000).

- QLEV=

  Mit QLEV kann eine zu starke Belastung des Pagepools durch Nachrichten für diese Temporäre Queue verhindert werden.
  QLEV legt den Standardwert für die maximale Anzahl von Nachrichten fest, die gleichzeitig in einer Temporären Queue stehen dürfen (Standardwert: 32767, d.h. unbeschränkte Queue-Länge).

- QMODE=

  Bestimmt das Verhalten von openUTM für den Fall, dass bereits die maximal erlaubte Anzahl von Nachrichten in einer Temporären Queue gespeichert und somit der Queue-Level erreicht ist.

5.3.4 Maximale Wartezeit beim Lesen aus Service-gesteuerten Queues festlegen

Bei der Generierung kann die maximale Zeitspanne festgelegt werden, die ein Vorgang auf das Eintreffen von Nachrichten für eine Queue wartet. Diese maximale Wartezeit kann jeweils für Dialog- und Asynchron-Vorgänge getrennt definiert werden (MAX-Anweisung, Operand QTIME). Dadurch wird z.B. verhindert, dass ein Terminalbenutzer oder Client wegen eines Fehlers in einem UTM-Teilprogramm minutenlang auf eine Reaktion des Systems warten muss oder Betriebsmittel deshalb zu lange belegt bleiben.

MAX-Anweisung auf Seite 368
Mit folgendem Operanden wird die Zeit festgelegt, die ein Vorgang maximal auf das Eintreffen von Nachrichten in einer Service-gesteuerten Queue warten darf:

- QTIME=

  Angabe der max. Wartezeit in Sekunden (Voreinstellung: 32767 Sekunden).
  Für das Warten in Dialog- bzw. Asynchron-Vorgängen kann jeweils ein eigener Maximalwert definiert werden.
  Wird in einem Teilprogrammlauf eine größere Wartezeit als bei QTIME angegeben, setzt openUTM die Wartezeit auf den hier definierten Wert zurück.

5.3.5 Maximale Anzahl der erneuten Zustellungen an Service-gesteuerte Queues begrenzen

Per Generierung kann festgelegt werden, ob eine Nachricht an eine Service-gesteuerte Queue erneut in die Queue gestellt werden soll, falls die Transaktion zurückgesetzt wird, in der diese Nachricht gelesen wurde. In der Generierung kann man die Anzahl dieser erneuten Zustellungen begrenzen (MAX-Anweisung, Operand REDELIVERY). Damit lassen sich z.B. Endlosschleifen bei Programmfehlern vermeiden.

MAX-Anweisung auf Seite 368
Mit folgendem Operanden wird die maximale Anzahl der erneuten Zustellungen von Nachrichten an Service-gesteuerte Queues festgelegt.

- REDELIVERY= (...,number2)

  number2 ist die maximale Anzahl wiederholter Zustellungen für Nachrichten an eine Service-gesteuerte Queue ($0 \leq number2 \leq 255$).

  Werte zwischen 0 und 254 geben die Anzahl an, der Wert 255 bedeutet, dass die Nachricht beliebig oft zugestellt werden kann.

  Voreinstellung 255, d.h. die Anzahl ist nicht begrenzt!

5.4 UTM-Meldungen

openUTM erzeugt beim Ablauf einer UTM-Anwendung Meldungen, die über bestimmte Ereignisse informieren oder Dialog-Eingaben anfordern. Die UTM-Meldungen sind in einem Meldungsmodul enthalten, das mit openUTM ausgeliefert wird (Standardmeldungsmodul).

Diese Meldungen von openUTM können Sie mit den Meldungstools KDCMTXT und KDCMMOD modifizieren und eigene Meldungsmodule (Benutzermeldungsmodule) erzeugen, die Ihren Anforderungen entsprechen. Sie können:

– Meldungstexte modifizieren (z.B. in andere Sprachen übersetzen),
– Meldungsziele löschen oder hinzufügen,
– Inserts ändern.

Benutzermeldungsmodule müssen Sie bei der KDCDEF-Generierung mit der MESSAGE-Anweisung definieren.

Im openUTM-Handbuch „Meldungen, Test und Diagnose“ sind das gesamte Meldungswesen und die Tools KDCMTXT und KDCMMOD detailliert beschrieben.

5.4.1 Meldungen in openUTM auf BS2000/OSD

B Folgende Bestandteile des UTM-Meldungswesens gehören zum Lieferumfang:

B
- das deutsche Standardmeldungsmodul KCSMSGS

B
- das englische Standardmeldungsmodul KCSMSGSE

B
- die Meldungsdefinitionsdatei SYSMSH.UTM.060.MSGFILE

B Die Meldungsdefinitionsdatei enthält die Meldungstexte in deutscher und englischer Sprache und bildet die Basis zur Erzeugung von Benutzermeldungsmodulen.

B Benutzermeldungsmodule müssen Sie mit der KDCDEF-Anweisung MESSAGE in die Konfiguration aufnehmen. Wird keine MESSAGE-Anweisung angegeben, dann wird das deutsche Standardmeldungsmodul KCSMSGS zur Ausgabe der Meldungen verwendet.

B Zur Internationalisierung der Anwendung können Sie mehrere Meldungsmodule in verschiedenen Sprachen erzeugen und in die Konfiguration einer UTM-Anwendung aufnehmen. Auf diese Weise können in einer UTM-Anwendung Meldungen in verschiedenen Sprachen an die Terminalbenutzer ausgegeben werden. Mit welcher Sprache ein Benutzer bedient wird, richtet sich nach dem Locale (Sprachkennzeichen *lang_id* und Territorialkennzeichen *terr_id*), das Sie ihm bei der Generierung zuordnen, sowie der Verfügbarkeit eines Benutzermeldungsmoduls, dem Sie bei der Generierung ein entsprechendes Locale zugeordnet haben.

B B Wird für eine UTM-Anwendung mehr als ein Meldungsmodul generiert, dann muss jedem Meldungsmodul ein Locale zugeordnet werden.

B B

MESSAGE-Anweisung auf Seite 407
Benutzermeldungsmodule definieren Sie mit den folgenden Operanden:

B
- MODULE=

B Name des Meldungsmoduls, das Sie in die Konfiguration aufnehmen wollen.

B
- LIB=

B B B B Bezeichnet die Objektmodulbibliothek, aus der das Meldungsmodul nachgeladen wird. Wenn ein generiertes Meldungsmodul *modulname* beim Binden der Anwendung nicht unter dem Namen *lmodname* in der Bibliothek *omlname* steht, meldet der Binder das Fehlen des Moduls. Das Meldungsmodul kann nachgeladen werden.

B
- LOCALE=

B B B B B B Sprachumgebung (Locale) des Meldungsmoduls festlegen, wenn landessprachliche Meldungsmodule für individuelle Meldungsausgaben erzeugt wurden. Diese landessprachlichen Meldungsmodule werden für Benutzer und LTERM-Partner eingesetzt, die in Sprach- und Territorialkennzeichen mit dem hier definierten Locale übereinstimmen. Weitere Informationen finden Sie im Abschnitt „Internationalisierung der Anwendung - XHCS Unterstützung (BS2000/OSD)“ auf Seite 235.

B B B

USER-Anweisung auf Seite 538 / **LTERM-Anweisung** auf Seite 354
Mit folgendem Operanden geben Sie das Meldungsmodul (und damit die Sprache) an, das für die Meldungsausgabe an den Benutzer/Client verwendet wird:

B
- LOCALE=

B Sprachumgebung (Locale) des Benutzers/Clients.

B *Anwendungsmeldungsmodul und Benutzermeldungsmodule*

B Werden für eine Anwendung mehrere Meldungsmodule eingesetzt, dann wird zwischen Anwendungsmeldungsmodul und Benutzermeldungsmodulen unterschieden. *Anwendungsmeldungsmodul* ist das Meldungsmodul, in dessen MESSAGE-Anweisung die Angaben für das Locale mit denen in der MAX-Anweisung übereinstimmen.
Dem Anwendungsmeldungsmodul kommt eine besondere Rolle in der Anwendung zu. Die für das Anwendungsmeldungsmodul gemachten Angaben zu den Meldungszielen bestimmen die Meldungsziele, an die eine Meldung ausgegeben wird. Die Angaben zu den Meldungszielen in den anderen Meldungsmodulen (Benutzermeldungsmodulen) sind ohne Bedeutung. Das Anwendungsmeldungsmodul wird zur Ausgabe von Meldungen an die Meldungsziele SYSLST, SYSOUT und CONSOLE verwendet.

B Für Meldungen an die Ziele STATION, SYSLINE und PARTNER wird das Meldungsmodul verwendet, dessen Angaben bei *lang_id* und *terr_id* (im Locale) mit denen des Benutzers bzw. LTERM-Partners übereinstimmen, für die die Meldung ausgegeben wird. Hierbei hat die Angabe beim Benutzer Vorrang vor der Angabe beim LTERM-Partner, d.h. ist zum Zeitpunkt der Meldungsausgabe ein Benutzer angemeldet, dann nimmt openUTM das zum Benutzer passende Meldungsmodul.

B Ist für einen Benutzer bzw. LTERM-Partner ein Locale (*lang_id*, *terr_id*) generiert, für das in der Anwendung kein Meldungsmodul existiert, dann wird dem Benutzer bzw. LTERM-Partner ein Meldungsmodul zugeordnet, das in *lang_id* übereinstimmt und für das keine *terr_id* generiert wurde. Ist auch kein solches Meldungsmodul vorhanden, dann wird zur Ausgabe von Meldungen an diesen Benutzer bzw. LTERM-Partner das Anwendungsmeldungsmodul verwendet.

B **MAX-Anweisung** auf Seite 368
Mit folgendem Operanden geben Sie an, welches Meldungsmodul Anwendungsmeldungsmodul ist:

B
- LOCALE=

  B Locale des Meldungsmoduls, das als Anwendungsmeldungsmodul verwendet werden soll. Ein Meldungsmodul mit diesem Locale muss mit einer MESSAGE-Anweisung generiert werden.

5.4.2 Meldungen in openUTM auf Unix- und Windows-Systemen

X/W Folgende Bestandteile des UTM-Meldungswesens gehören zum Lieferumfang:

X/W
- Standardmeldungsmodul von openUTM

X/W Das Standardmeldungsmodul enthält die Texte der Standardmeldungen in englischer
X/W Sprache und Standardeinstellungen für die Meldungsziele (z.B. Terminals, SYSLOG-
X/W Datei).
X/W Nur falls für eine Sprache keine NLS-Meldungskataloge und keine Benutzer-spezifi-
X/W schen Meldungsmodule vorhanden sind, erzeugt openUTM die Meldungen aus dem
X/W Standardmeldungsmodul.

X/W

VORSICHT!
Das Standardmeldungsmodul muss in **jedes** UTM-Anwendungsprogramm eingebunden werden.

Unix-Systeme

X Mit openUTM auf Unix-Systemen werden die Module *kcsmsgs.o* (K- und P-Meldungen)
X und *kcxmsgs.o* (U-Meldungen) ausgeliefert. Beide sind in der Bibliothek
X *utmpfad*`/sys/libwork.a` bzw.
X *utmpfad*`/sys/libwork.so` enthalten.

X Für beide Module wird der Ausdruck „Standardmeldungsmodul“ verwendet.

W *Windows-Systeme*

W Das Standardmeldungsmodul wird im Objekt *utmpfad\sys\kcsmsgs.obj* ausgeliefert.

X/W
- Meldungsdefinitionsdatei *msgdescription (*im *utmpfad*)

X/W Sie enthält die Standardmeldungstexte in deutscher und englischer Sprache sowie die
X/W Rahmendefinitionen (Struktur) für die Meldungen.

X/W
- NLS-Standard-Meldungskataloge (Unix-Systeme) / Meldungs-DLLs (Windows-Syste-
X/W me)

X/W Mit openUTM werden NLS-Standard-Meldungskataloge in deutscher und englischer
X/W Sprache ausgeliefert. Unter Windows-Systemen sind die Meldungskataloge als Mel-
X/W dungs-DLLs realisiert.

X/W Die Meldungskataloge enthalten nur die Meldungstexte. Beim Aufbau der Meldungen
X/W aus einem NLS-Katalog entnimmt openUTM die Strukturinformationen und die Mel-
X/W dungsziele dem Standardmeldungsmodul bzw. - wenn vorhanden - einem Benutzer-
X/W meldungsmodul.

X *Unix-Systeme*

X Die NLS-Standard-Meldungskataloge werden auf Unix-Systemen in den Dateiverzeichnissen *utmpfad*/nls/msg/*xxx* abgelegt. Dabei ist *xxx* das Kennzeichen der jeweiligen Sprache.

X Auf Unix-Systemen können Sie die vorhandenen NLS-Meldungskataloge modifizieren und eigene NLS-Meldungskataloge für andere Sprachen erzeugen.

X Mit der Shell-Variablen LANG können Sie Ihre bevorzugte Meldungssprache Benutzerspezifisch festlegen.

W *Windows-Systeme*

W Unter Windows-Systemen finden Sie die Meldungs-DLLs in den Dateiverzeichnissen *utmpfad*\nls\msg*xxx*. Dabei ist *xxx* das Kennzeichen der jeweiligen Sprache.

W Auf Windows-Systemen können Sie mit einem eigenen Meldungsmodul die Meldungsziele ändern, nicht jedoch die Texte der Meldungen.

W Mit der Shell-Variablen LANG können Sie Ihre bevorzugte Meldungssprache Benutzerspezifisch festlegen.

X/W Im einfachsten Fall betreiben Sie Ihre Anwendung mit den UTM-Standardmeldungen, d.h. Sie nehmen keine Änderungen an den Meldungen und den Meldungszielen vor. In diesem Fall sind keine zusätzlichen Angaben bei der KDCDEF-Generierung erforderlich. Sie müssen nur das Standardmeldungsmodul zum Anwendungsprogramm binden.

X/W Beim Generieren einer Anwendung wird mit der Anweisung MESSAGE der Name des Meldungsmoduls definiert. Dieses Meldungsmodul wird aus einer von KDCMMOD geschriebenen C-Quelldatei erzeugt.

X/W

MESSAGE-Anweisung auf Seite 407
Mit folgendem Operanden wird beim Generieren der Anwendung das Meldungsmodul definiert:

X/W
- MODULE=

  X/W Name des Moduls, das mit dem Tool KDCMMOD erzeugt wird.

X/W Zum Modifizieren von Standardmeldungen verwenden Sie die Meldungstools KDCMTXT und KDCMMOD (siehe openUTM-Handbuch „Meldungen, Test und Diagnose“).

5.4.3 Benutzer-spezifische Meldungsziele

Zusätzlich zu den Meldungszielen CONSOLE, SYSOUT, usw. gibt es vier weitere, so genannte Benutzer-spezifische Meldungsziele. Mit ihnen kann der Benutzer bis zu vier eigene Meldungsziele vereinbaren. Diese Meldungsziele werden USER-DEST-*number* genannt und können User-Queues, TAC-Queues, Asynchron-TACs oder LTERM-Partner sein.

Damit ist es u.a. möglich, K- und P-Meldungen Ihrer Anwendung einem Administrator am WinAdmin-Administrationsarbeitsplatz anzuzeigen (siehe auch „WinAdmin Online-Hilfe“).

Die Meldungen, die das Über- bzw. Unterschreiten des Warnlevels anzeigen, können jedoch nicht in allen Fällen dem Benutzer-spezifischen Meldungsziel zugestellt werden.

Für die Vereinbarung Benutzer-spezifischer Meldungsziele wird die KDCDEF-Anweisung MSG-DEST verwendet.

MSG-DEST-Anweisung auf Seite 414
Mit den folgenden Operanden können max. vier Benutzer-spezifische Meldungsziele vereinbart werden:

- *msg-destination*

  Bezeichnet das Meldungsziel mit der Angabe USER-DEST-*number (number*=1..4*)*. Meldungsziele müssen den Meldungen mit KDCMMOD zugeordnet werden.

- NAME=

  Gibt den Namen einer User- oder TAC-Queue oder eines Asynchron-TACs oder LTERM-Partners an, an die/den die Meldungen gesendet werden sollen (dieser Name muss mit einer TAC-, USER- oder LTERM-Anweisung definiert werden).

- DEST-TYPE=

  Typ des Meldungsziels (USER-Queue, TAC oder LTERM).

- MSG-FORMAT=

  Legt das Format der zu sendenden Meldungen fest. Es werden entweder nur die Inserts im nicht-abdruckbaren Format (FILE; Voreinstellung) oder Inserts und Meldungstext im abdruckbaren Format (PRINT) übergeben.

Meldungsziel USER-DEST-*number* zuordnen

Den Meldungen, die openUTM an eines der Meldungsziele USER-DEST-*number* ausgeben soll, muss zusätzlich mit dem Dienstprogramm KDCMMOD (Anweisung MODMSG) dieses Meldungsziel zugeordnet werden.

5.5 Nachrichtenverteilung und Multiplexing mit OMNIS (BS2000/OSD)

Für UTM-Anwendungen in BS2000/OSD lassen sich die Dienste des BS2000/OSD-Softwareprodukts OMNIS nutzen. OMNIS ist ein Session Manager, der es ermöglicht, dass ein Terminalbenutzer Services verschiedener UTM-Anwendungen direkt aufrufen kann - selbst dann, wenn die UTM-Anwendungen im Netz verteilt sind. Dabei muss der Terminal-Benutzer nicht wissen, auf welchem Rechner und in welcher UTM-Anwendung der Service liegt: OMNIS baut automatisch eine Verbindung zur „richtigen“ UTM-Anwendung auf und regelt die Zuordnung der Nachrichten (Nachrichtenverteilung).

Zusätzlich können Sie beim Einsatz von OMNIS die Multiplexfunktion nutzen, die openUTM in BS2000/OSD zur Verfügung stellt: Eine große Zahl von Terminals kann über eine kleine Zahl von Transportverbindungen mit einer UTM-Anwendung in Verbindung stehen.

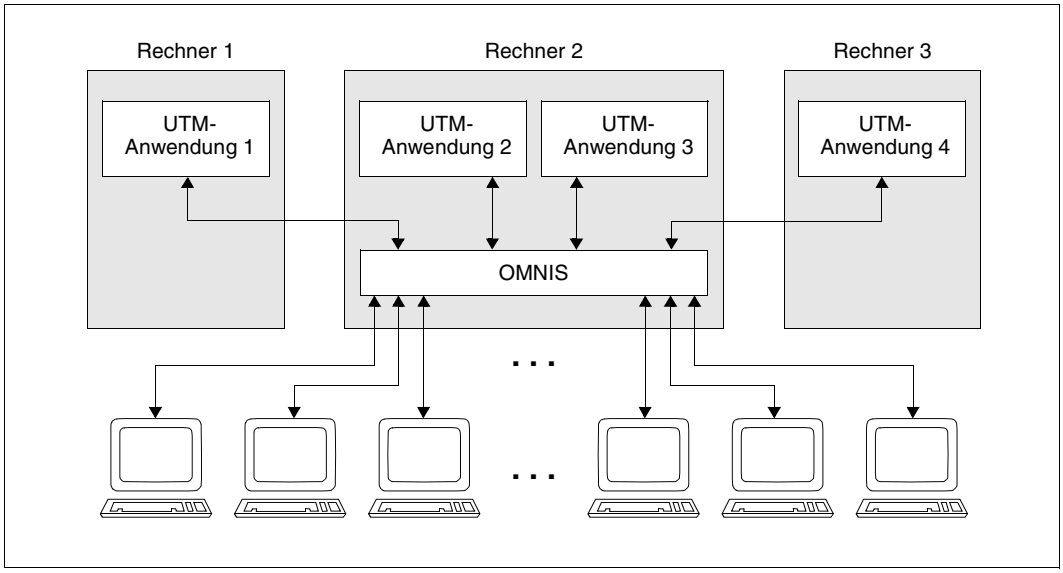


Bild 15: Nachrichtenverteilung und Multiplexing mit OMNIS

Sehen Sie dazu auch die Handbücher „OMNIS/OMNIS-MENU Funktionen und Kommandos“ und „OMNIS/OMNIS-MENU Administration und Programmierung“.

OMNIS unterstützt ab V6.3A XHCS (Extended Host Code Support, siehe Abschnitt „Internationalisierung der Anwendung - XHCS Unterstützung (BS2000/OSD)“ auf Seite 235), ab OMNIS V7.0A auch bei UTM-Partnern mit Multiplexprotokoll. Voraussetzung für die Verwendung von XHCS ist, dass OMNIS auf einem Rechner mit DCAM ≥ V10.0 und VTSU-B ≥ V10.1B läuft.

5.5.1 Multiplexverbindungen

Im normalen Dialog-Betrieb besteht zwischen einem Terminal und einer UTM-Anwendung auf dem Rechner jeweils eine Transportverbindung. Damit ein Benutzer Services einer Anwendung aufrufen kann, muss er eine Session zwischen sich und der Anwendung aufbauen, d.h. eine Kommunikationsbeziehung zwischen zwei adressierbaren Einheiten im Netz. Ein Sessionaufbau beinhaltet im Allgemeinen, dass sich der Benutzer gegenüber der Anwendung identifizieren muss. Dies kann auch implizit geschehen.

OMNIS bietet dem Benutzer nun die Möglichkeit, gleichzeitig mit mehreren UTM-Anwendungen auch auf verschiedenen Rechnern in Verbindung zu treten. Der Benutzer ist aber tatsächlich nur mit einem Kommunikationspartner (nämlich OMNIS) verbunden. Der Session Manager vermittelt nun die Eingabe-Nachrichten (Aufträge des Benutzers) an die Anwendungen, mit denen der Benutzer in Verbindung steht.

Auf beiden Teilstrecken der Kommunikation, d.h. Strecken Benutzer ⟶ Session Manager und Session Manager ⟶ Anwendung, bestehen Transportverbindungen und Sessions. Die folgende Abbildung macht dies deutlich:

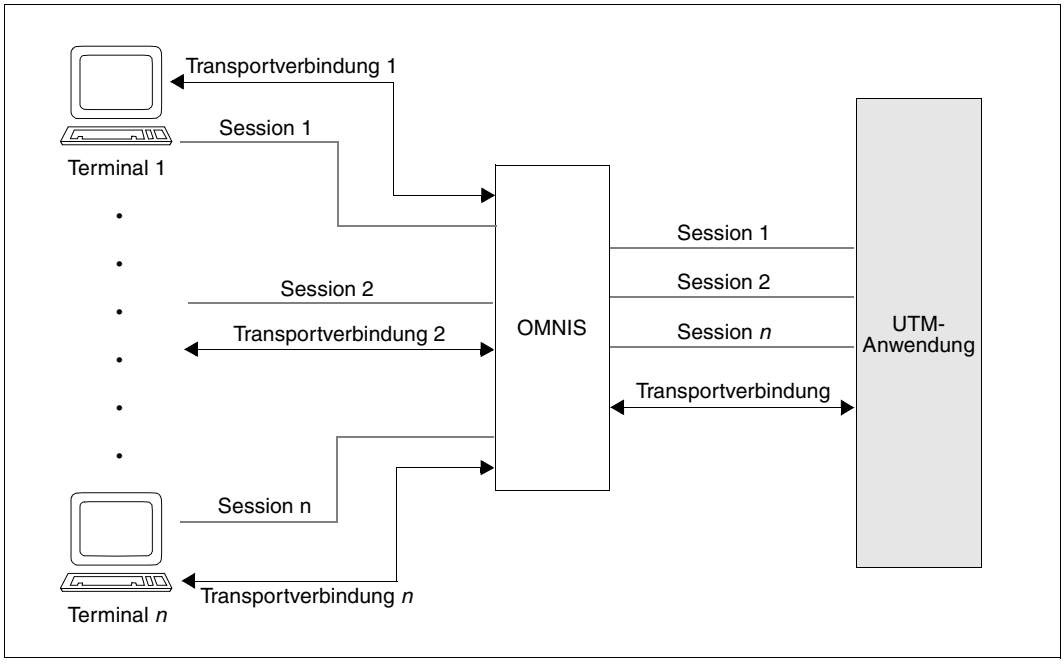


Bild 16: Transportverbindungen und Sessions beim Multiplexen

Eine **Transportverbindung** ist eine Verbindung zwischen zwei Programmen bzw. einem Programm und einem Terminal, über die Nachrichten ausgetauscht werden können. Eine Transportverbindung hat einen definierten Anfang (Verbindungsaufbau) und ein definiertes Ende (Verbindungsabbau) und ist dem Transportsystem bekannt.

Eine **Session** ist einer von mehreren wohl unterschiedenen Datenströmen, die über eine Transportverbindung geführt werden. Eine Session hat einen definierten Anfang (Session-Aufbau) und ein definiertes Ende (Session-Abbau) und ist dem Transportsystem nicht bekannt. Im speziellen Fall von OMNIS und openUTM versteht man unter einer Session eine Kommunikationsbeziehung zwischen einer UTM-Anwendung und einem OMNIS-Terminal, die mit dem logischen Aufbau der Session beginnt und mit deren Abbau endet.

Auf der Teilstrecke Terminal ⟶ Session Manager besteht eine eins-zu-eins Zuordnung zwischen Transportverbindung und Session.

Auf der Teilstrecke Session Manager ⟶ Anwendung wird diese eins-zu-eins Zuordnung aufgehoben und einer Transportverbindung können mehrere Sessions zugeordnet werden. Auf diese Weise können mehrere Terminals „multiplexen“, d.h. über eine Transportverbindung an eine Anwendung angeschlossen werden. Im Extremfall können sogar alle Sessions zwischen Session Manager und Anwendung über eine einzige Transportverbindung abgewickelt werden.

5.5.1.1 Multiplexverbindungen definieren

Jede Multiplexverbindung muss mit einer MUX-Anweisung beschrieben werden. Es ist nicht möglich, Multiplexverbindungen dynamisch einzutragen.

Die Kommunikation zwischen Session Manager und Anwendung erfolgt beim Multiplexen über das PUTMMUX-Protokoll. Die Aufgabe dieses Protokolls ist es, das Abwickeln mehrerer Sessions über eine Transportverbindung zu ermöglichen und dem Session Manager Statusinformationen über die UTM-Anwendung unter BS2000/OSD zur Verfügung zu stellen.

Eine PUTMMUX-Verbindung kann zwischen einer UTM-Anwendung unter BS2000/OSD und OMNIS als Session Manager bestehen. PUTMMUX-Verbindungen, auch „Multiplexverbindungen“ genannt, werden beim Generieren der UTM-Anwendung durch die MUX-Anweisung definiert.

MUX-Anweisung auf Seite 417

Die wichtigsten Eigenschaften für Multiplexverbindungen werden mit den folgenden Operanden festgelegt:

- name

  Name der Multiplexverbindung.

- BCAMAPPL=

  Lokaler Anwendungsname der UTM-Anwendung, über den der Session Manager die Verbindung zur UTM-Anwendung aufbaut.

- CONNECT=

  Aufbau einer Transportverbindung zum Session Manager beim Anwendungsstart.

- MAXSES=

  Maximale Anzahl gleichzeitig aktiver Sessions zwischen Session Manager und UTM-Anwendung.

Beim Aufbau einer Multiplexverbindung handeln openUTM und OMNIS aus, welche MUX-Protokollversionen von beiden Seiten der Verbindung unterstützt werden. Gibt es keine MUX-Protokollversion, die von beiden Partnern unterstützt wird, wird die Multiplexverbindung nicht aufgebaut (Meldung K140 und K141).

Bei der jetzigen Auslegung des Protokolls gelten folgende Einschränkungen:

- Verbindungen zwischen zwei UTM-Anwendungen werden nicht unterstützt
- Drucker werden nicht unterstützt
- Ein Sessionaufbau zur UTM-Anwendung kann nur vom Session Manager erfolgen

Für die menügeführte Nutzung von OMNIS steht Ihnen das Zusatzprodukt OMNIS-MENU zur Verfügung. OMNIS-MENU ermöglicht dem Benutzer, über eine komfortable, menügesteuerte Oberfläche mit verschiedenen UTM-Anwendungen zu kommunizieren. Nähere Informationen sind den Handbüchern „OMNIS/OMNIS-MENU Funktionen und Kommandos“ und „OMNIS/OMNIS-MENU Administration und Programmierung“ zu entnehmen.

5.5.1.2 Asynchron-Nachrichten ohne Vorankündigung ausgeben

B Ab OMNIS V7.0 kann openUTM asynchrone Nachrichten auch ohne Vorankündigung an
B Terminals senden, die über eine Multiplexverbindung mit der Anwendung verbunden sind.
B D.h. openUTM kündigt eine für den Benutzer anstehende asynchrone Nachricht nicht mit
B der Meldung K012 an, sondern übergibt die Nachricht sofort an OMNIS. OMNIS sorgt in
B diesem Fall für die Ausgabe der asynchronen Nachricht.

B Damit können auch Terminals, die sich über eine Multiplexverbindung an eine UTM-Anwen-
B dung anschließen, Sessions über LTERM-Partner aufbauen, die durch eine LTERM-Anwei-
B sung bzw. TPOOL-Anweisung mit ANNOAMSG=N generiert sind.

5.5.1.3 Bestätigung des Verbindungsabbaus durch den Partner

B Ist ein Benutzer über eine Multiplexverbindung mit einer UTM-Anwendung verbunden, dann
B kann ab OMNIS V7.0A jeder der beiden Partner, die UTM-Anwendung oder der Benutzer,
B den Abbau dieser Session anfordern. Die Session geht durch die Anforderung in den Zu-
B stand „DISCONNECT PENDING“ über. Sie wird noch nicht freigegeben. Der endgültige Ab-
B bau der Session erfolgt erst, wenn der Partner auf der anderen Seite den Sessionabbau
B bestätigt hat.

B Für eine bestimmte Zeitspanne (ca. 10 Minuten) nach der Aufforderung zum Sessionabbau
B kann die Session nur durch die Abbaubestätigung des Partners freigegeben werden.
B Erst nach Ablauf der Zeitspanne kann auch der Administrator der UTM-Anwendung die
B Session freigeben (Administrationskommando KDCPTERM).

B Den Ausgaben der Administrationskommandos KDCINF PTERM und KDCPTERM kann
B der Administrator der UTM-Anwendung entnehmen, ob sich eine Session im Zustand
B „DISCONNECT PENDING“ befindet. Sehen Sie dazu auch das openUTM-Handbuch „An-
B wendungen administrieren“.

5.5.2 Statistiken zu Multiplexverbindungen

Der Administrator der UTM-Anwendung kann sich mit Hilfe des Kommandos

```
KDCINF MUX,OPTION=MONITORING
```

von openUTM Statistiken über Multiplexanschlüsse ausgeben lassen. Sehen Sie dazu auch das openUTM-Handbuch „Anwendungen administrieren“. Der UTM-Administrator wird informiert über:

- die Auslastung der Multiplexverbindung

  Es werden Informationen darüber geliefert, wieviele Ein- und Ausgabe-Nachrichten seit dem Start der Anwendung über Multiplexverbindungen ausgetauscht wurden.

- BCAM-Engpässe

  openUTM liefert Information darüber, wieviele Nachrichten der Anwendung BCAM seit dem Anwendungsstart auf Grund von BCAM-Engpässen nicht annehmen konnte und deshalb openUTM zum erneuten Senden einer Nachricht auffordern musste.

5.5.3 Kombination von Multiplexverbindungen und direkten Verbindungen

Wenn Sie sowohl Terminals über direkte Verbindungen als auch Terminals über Multiplexverbindungen des Session Managers an Ihre UTM-Anwendung anschließen, ergibt sich die folgende Verteilung.

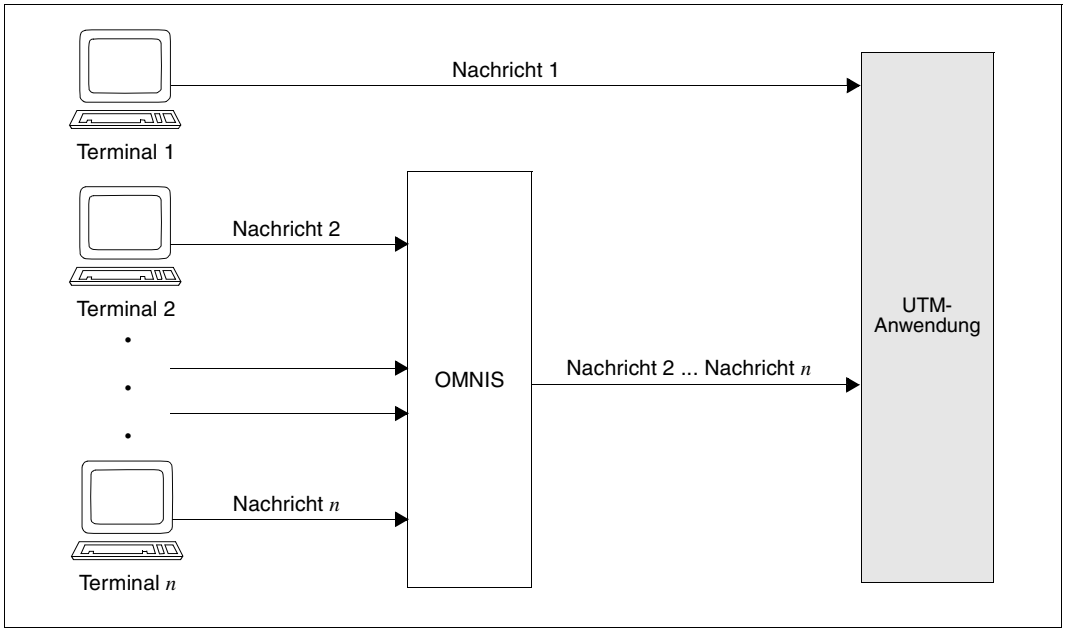


Bild 17: Kombination von Multiplex- und direkten Verbindungen

Dabei können Nachrichten über direkte Verbindungen Nachrichten über Multiplexverbindungen überholen. Das führt in bestimmten Lastsituationen zu kürzeren Antwortzeiten auf der direkten Verbindung und zwar dann, wenn es zu einem Datenstau auf den Multiplexverbindungen kommt. Dafür kann es mehrere Gründe geben:

- das Nachrichten-Aufkommen von den Terminals ist so hoch, dass die Multiplexverbindungen überlastet sind.
- alle UTM-Prozesse sind mit Aufträgen belegt und können daher nicht alle ankommenden Nachrichten sofort abholen.

Für den UTM-Administrator der betroffenen UTM-Anwendung gibt es zwei Möglichkeiten, die Wahrscheinlichkeit eines Datenstaus zu vermindern:

- die Anzahl der Multiplexverbindungen erhöhen und das Nachrichten-Aufkommen gleichmäßig über diese verteilen
- die aktuelle Anzahl der UTM-Prozesse erhöhen

Um dem Administrator immer schnellstmöglichen Zugriff auf die UTM-Anwendung zu sichern, sollte sein Terminal über eine direkte Verbindung an die Anwendung angeschlossen werden.

5.6 Lademodule, Common Memory Pools und Shared Code generieren (BS2000/OSD)

In diesem Abschnitt wird beschrieben, wie Sie Teilprogramme, Areas und Lademodule generieren müssen.

Im openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“ finden Sie weitere Informationen und Empfehlungen

- zur Strukturierung eines Anwendungsprogramms
- zur Bereitstellung von shared Code in Systemspeicher oder Common Memory Pools
- zur Reihenfolge, in der die Module geladen werden und wie dabei die Externverweise aufgelöst werden
- zum Programmaustausch im laufenden Betrieb

5.6.1 Lademodule generieren

Nur ein Teil der Anwendung muss statisch zum Anwendungsprogramm gebunden werden (Start-LLM, siehe openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“, Binden mit Binder). Die anderen Teile des Anwendungsprogramms können Sie in Form von dynamisch nachladbaren Lademodulen bereitstellen.

Bereits bei der KDCDEF-Generierung legen Sie fest, zu welchem Zeitpunkt die Anwendungsteile, die nicht statisch gebunden werden, nachgeladen werden und in welchen Speicherbereich sie geladen werden. Darüber hinaus legen Sie fest, welche Programmteile später im laufenden Betrieb austauschbar sein sollen.

Sie müssen für jedes Lademodul der Anwendung, das nicht zum statisch gebundenen Teil der Anwendung gehört, genau eine LOAD-MODULE-Anweisung absetzen. Dabei geben Sie an, wann das Lademodul dazugebunden und wohin es geladen werden soll. Die Reihenfolge der LOAD-MODULE-Anweisungen gibt die Reihenfolge an, in der die Lademodule geladen werden (siehe Beschreibung der LOAD-MODULE-Anweisung auf Seite 336 und im openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“, Module laden).

Die Zuordnung von nicht statisch gebundenen Objekten (Teilprogramme und Areas) zu den Lademodulen wird ebenfalls bei der Generierung festgelegt. In den PROGRAM- und AREA-Anweisungen, in denen Teilprogramme bzw. Areas generiert werden, geben Sie dazu im Operand LOAD-MODULE den Namen des Lademoduls an, das dieses Teilprogramm bzw. die Area enthält und das Sie mit einer LOAD-MODULE-Anweisung generiert haben.

B B B B B B B B B B B B B

VORSICHT!
Ob die mit der Anweisung LOAD-MODULE und den Operanden LOAD-MODULE in der PROGRAM- und AREA-Anweisung definierte Zuordnung der tatsächlichen Aufteilung der Lademodule in den Bibliotheken entspricht, kann von openUTM nicht verifiziert werden.
openUTM verlässt sich beim Nachladen der Lademodule und beim Programmaustausch auf die in der Generierung gemachten Angaben. Sie müssen deshalb sicherstellen, dass die von Ihnen verwendeten Bindeprozeduren für die einzelnen Teile des Anwendungsprogramms mit den in der Generierung gemachten Angaben übereinstimmen. Andernfalls kann openUTM nicht sicherstellen, dass mit einem bestimmten Lademodul das gewünschte Programm in den Arbeitsspeicher geladen wird.

B Die Lademodule werden bei der Generierung wie folgt beschrieben:

B B B

LOAD-MODULE-Anweisung auf Seite 336
Die Eigenschaften der Lademodule werden mit den folgenden Operanden festgelegt:

B

- *lmodname*

B B

Name des Lademoduls. Über diesen Namen erfolgt bei der Generierung die Zuordnung der Objekte (Teilprogramme, Areas) zu den Lademodulen.

B B

Sie dürfen nur die Namen von OMs oder LLMs angeben. Aus Performancegründen unterstützt openUTM nicht das Nachladen über CSECT- oder ENTRY-Namen.

B

- LOAD-MODE=

B B B B

gibt an, wann ein Lademodul geladen werden soll, und legt den Speicherbereich fest, in den das Lademodul geladen werden soll. Die Lademodule können in den Standardkontext im tasklokalen Speicher, in einen Common Memory Pool oder in den Systemspeicher geladen werden.

B

Die Teile des Anwendungsprogramms können:

B B

– statisch zum Anwendungsprogramm gebunden werden (LOAD-MODE=STATIC)

B B B

Das ist der Teil des Anwendungsprogramms, der mit dem Kommando `START-EXECUTABLE-PROGRAM` bzw. `LOAD-EXECUTABLE-PROGRAM` in den Standardkontext der Anwendung geladen wird.

– beim Start der Anwendung dynamisch in den Standardkontext des tasklokalen Speichers nachgeladen werden (LOAD-MODE=STARTUP).

  Das sollten Programmteile sein, die ständig von der UTM-Anwendung benötigt werden oder Externverweise zu shareable Teilen der Anwendung enthalten.

– erst beim ersten Aufruf in den Standardkontext des tasklokalen Speichers geladen werden (LOAD-MODE=ONCALL)

  Das sollten Programmteile sein, die von der Anwendung nicht ständig benötigt werden.

– in einen Common Memory Pool geladen werden (LOAD-MODE=(POOL,*poolname,*...))

  Der Common Memory Pool muss mit einer MPOOL-Anweisung generiert werden, siehe dazu Seite 202.
  In einen Common Memory Pool sollten Sie Programmteile laden, die von allen Prozessen einer UTM-Anwendung gemeinsam benutzt werden können, wie z.B. die shareable Teile Ihrer Teilprogramme oder auch Formate oder Datenbereiche.
  Enthält ein LLM Public und Private Slices, dann wird die Public Slice in einen Common Memory Pool und die Private Slice in den Standardkontext im tasklokalen Speicher geladen. Dabei können Sie angeben, ob der non-shareable Teil beim Start der Anwendung nachgeladen werden soll (LOAD-MODE=(POOL, poolname, STARTUP)) oder erst zum Zeitpunkt des Teilprogrammaufrufs (LOAD-MODE=(POOL, poolname, ONCALL)). Zum Generieren von shared Code siehe auch Seite 201ff.

– als nicht-privilegiertes Subsystem in den Systemspeicher geladen werden.

  Diese Anwendungsteile müssen vor dem Start der Anwendung vom BS2000-Systemverwalter in den Systemspeicher geladen werden.

  Die Private Slice zu einem shareable Teil, der sich in nicht-privilegierten Subsystemen befindet, kann man zum statischen Teil des Anwendungsprogramms binden, beim Start der Anwendung oder erst beim ersten Aufruf nachladen. Wie Sie den nicht-shareablen Teil generieren müssen ist auf Seite 201 beschrieben.

- LIB=

  gibt an, aus welcher Bibliothek das Lademodul geladen werden soll

  Sie können Bindemodulbibliotheken (OML) oder Programmbibliotheken (PL), die Elemente vom Typ R oder L enthalten, angeben.

- VERSION=

  gibt an, welche Version eines Lademoduls geladen werden soll

  In einer Programmbibliothek können gleichzeitig mehrere Versionen eines Elements enthalten sein. Mit der Versionsnummer legen Sie fest, welche Version eines Elements geladen werden soll.

- ALTERNATE-LIBRARIES=

  gibt an, ob mit Autolink gebunden werden soll.

  Die shareable Teile der Lademodule werden stets ohne die Autolink-Funktion geladen. Bei den nicht-shareable Teilen können Sie über die ALTERNATE-LIBRARIES= steuern, ob mit oder ohne Autolink-Funktion geladen werden soll.

  Geben Sie ALTERNATE-LIBRARIES=NO an, dann unterdrückt openUTM beim Nachladen und beim Programmaustausch die Autolink-Funktion des BLS. Das Lademodul darf ausschließlich offene Externverweise auf Programmteile besitzen, die zum Zeitpunkt des Ladens dieses Moduls bereits im Arbeitsspeicher vorhanden sind.
  Bei Lademodulen, die mit POOL oder STARTUP generiert werden, ist die Reihenfolge der LOAD-MODULE-Anweisungen bei der Generierung ausschlaggebend für die Auflösung der unbefriedigten Externverweise beim Laden. Die Reihenfolge der LOAD-MODULE-Anweisungen gibt die Reihenfolge an, in der die Lademodule geladen werden.

  ALTERNATE-LIBRARIES=YES bewirkt, dass beim Laden die Module, die zusätzlich benötigt werden, nachgebunden werden. Die Autolink-Funktion darf nur für Module des Laufzeitsystems, *nicht* jedoch für benutzereigene Module verwendet werden, da per Autolink geladene Module bei einem nachfolgenden Austausch nicht mehr entladen werden.

Module, die weder Teilprogramme des Anwendungsprogramms noch Areas sind (z.B. die Module der Laufzeitsysteme der Programmiersprachen), brauchen Sie, selbst wenn diese Module nicht statisch eingebunden sind, nicht einzeln mit dem Generierungstool KDCDEF als nachzuladende Module zu deklarieren. Sie können diese Module zu größeren Lademodulen (LLM) vorbinden und müssen nur den Namen des Lademoduls in der LOAD-MODULE-Anweisung generieren.

Die Zuordnung der Teilprogramme und Areas zu den Lademodulen wird bei der Generierung wie folgt festgelegt:

AREA-Anweisung auf Seite 286, **PROGRAM-Anweisung** auf Seite 437
Die Zuordnung zu einem Lademodule wird mit folgendem Operanden festgelegt:

- LOAD-MODULE=

  Name des Lademoduls (*lmodname* aus der LOAD-MODULE-Anweisung), zu dem das Programm gehört.

  Teilprogramme, Module und Datenbereiche müssen statisch zum Anwendungsprogramm gebunden werden, wenn sie keinem Lademodul zugeordnet sind.

  Die Administrationsmodule (z.B. das Administrationsprogramm KDCADM) sollten entweder statisch zum Start-LLM oder zu einem eigenen Lademodul gebunden werden. Dieses Lademodul muss beim Start der Anwendung geladen werden (LOAD-MODE=STARTUP). Dasselbe gilt für die Event-Exits START, SHUT, INPUT und FORMAT und die Event-Services BADTAC, MSGTAC und SIGNON.

Wenn bei der Generierung Angaben zu Objekten in den Anweisungen AREA, LOAD-MODULE, MPOOL, PROGRAM und TAC geändert werden, muss nur eine neue KDCFILE erzeugt werden. Der nächste Start der Anwendung muss dann mit der neuen KDCFILE erfolgen.

5.6.2 Shared Code und Common Memory Pools generieren

B Viele Compiler bieten eine Option an, die es gestattet, bei der Übersetzung von Programmen einen shareable Teil zu erzeugen. Dieser muss nicht unbedingt in einem eigenen Objektmodul abgelegt werden, sondern kann zusammen mit dem nicht-shareable Teil in einem LLM stehen, der in eine Public und eine Private Slice unterteilt ist.

B Wenn Teile eines Teilprogramms shareable sein sollen, so müssen Sie das bereits beim Programmieren berücksichtigen (siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“).

5.6.2.1 Shared Code im Systemspeicher

B **Shareable Teile** der Anwendungsteilprogramme und Teile der Laufzeitsysteme können mit Hilfe der im BS2000/OSD vorgesehenen Schnittstellen als shareable Programme in nichtprivilegierte Subsysteme geladen werden.

B Die shareable Module müssen durch den Administrator in den Speicher geladen werden, bevor die Anwendung gestartet wird. Sie können während des Betriebs der Anwendung ausgetauscht werden.

B Die **non-shareable Teile** der Teilprogramme müssen Sie wie folgt generieren:

B
- Der Einsprungpunkt des Teilprogramms (er liegt im non-shareable Teil oder in der Private Slice) muss in einer PROGRAM-Anweisung beschrieben und dort über den Operanden LOAD-MODULE einem Lademodul zugeordnet werden.

B
- Das Lademodul müssen Sie mit einer LOAD-MODULE-Anweisung mit LOAD-MODE= {STARTUP | ONCALL} generieren. Das Lademodul bzw. dessen private Slice wird beim Start des Anwendungsprogramms dynamisch in den tasklokalen Speicher (Klasse-6-Speicher) geladen. Über die Externbezüge zu den shareable Modulen werden die Verknüpfungen in den Shared Code dynamisch hergestellt.

B Die Lademodule (OM-Format), die die shareable Module des Teilprogramms enthalten, müssen nicht zusammen mit den Lademodulen, die die nicht-shareable Programmteile enthalten, in einer Programmbibliothek stehen.

B *Beispiel*

B
```
PROGRAM NONSHARE,LOAD-MODULE=NAME1,COMP=ILCS
LOAD-MODULE NAME1,LIB=UTM.PLIB,LOAD-MODE=STARTUP,VERSION=001
```

B NONSHARE befindet sich im nicht-shareable Teil (für LLMs im Private Slice) des Teilprogramms.

5.6.2.2 Shared Code im Common Memory Pool

In einen Common Memory Pool können Sie Objekte laden, die beim Binden des Anwendungsprogramms nicht statisch dazugebunden wurden. In einen Common Memory Pool können Sie mehrere Lademodule dynamisch laden.
Einen Common Memory Pool müssen Sie mit der KDCDEF-Anweisung MPOOL generieren.

MPOOL-Anweisung auf Seite 411
Die wichtigsten Eigenschaften eines Common Memory Pools legen Sie mit den folgenden Operanden fest:

- *poolname*

  Namen des Common Memory Pools. Über diesen Namen ordnen Sie bei der Generierung dem Pool die Lademodule zu, deren Public Slice in den Pool geladen werden soll (siehe LOAD-MODULE-Anweisung).

- SCOPE=

  legt den Geltungsbereich des Pools fest (anwendungslokal mit SCOPE=GROUP oder anwendungsglobal mit SCOPE=GLOBAL).
  BLS unterstützt pro BS2000-Benutzerkennung maximal acht Common Memory Pools mit SCOPE=GROUP und acht Common Memory Pools mit SCOPE=GLOBAL.

- PAGE=

  Hexadezimale Adresse in der Form X'xxxxxxxx'.
  Werden globale Common Memory Pools gleichen Inhalts/Namens in mehreren UTM-Anwendungen verwendet, muss der Parameter PAGE=X'xxxxxxxx' mit gleicher Adresse in allen Anwendungen angegeben werden. Die mit PAGE= angegebene Adresse ist so zu wählen, dass der damit reservierte Adressbereich in all diesen Anwendungen verfügbar ist.

- SIZE=

  legt die Größe des Common Memory Pools fest.
  Die Größe wird in Einheiten von 64 KB angegeben. Sie ist bei 24-Bit-Adressierung immer ein Vielfaches von 64 KB. Bei 31-Bit-Adressierung errechnet sich die Größe des Common Memory Pools durch $n * 1\text{MB} \geq \text{SIZE} * 64\text{ KB}$ (wobei n minimal gewählt wird).

Es sollte nur **ein** Common Memory Pool mit SCOPE=GROUP definiert werden. In diesen können Sie mehrere vorgebundene Lademodule laden. Damit wird die Zeit zum Einrichten und Laden der Common Memory Pools verkürzt und somit die für den Start der Anwendung benötigte Zeit minimiert.

B **Shareable Objekte generieren, die in einen Common Memory Pool geladen werden**

B Im folgenden wird beschrieben, wie Sie shareable Objekte generieren, die in einen Com-
B mon Memory Pool geladen werden sollen.

- B Aus Performancegründen sollten Sie, soweit möglich, alle shareable Teile eines
B Anwendungsprogramms, die in einen Common Memory Pool geladen werden sollen,
B zu **einem** Lademodul zusammenfassen.

- B Das vom Compiler erzeugte shareable Code-Modul des Programms muss in einem
B LLM oder OM enthalten sein. LLMs mit Slices können Sie mit einer einzigen LOAD-
B MODULE-Anweisung generieren:

```
B LOAD-MODULE llm-name ,VERSION=version -
B ,LOAD-MODE=(POOL,poolname,{STARTUP|ONCALL})-
B ,LIB=program-lib-
B .ALTERNATE-LIBRARIES={YES|NO}
```

B Durch diese Anweisung wird die Public Slice des LLM in den Common Memory Pool
B *poolname* geladen, die Private Slice wird entweder bei Anwendungsstart (STARTUP)
B oder bei Programmaufruf (ONCALL) nachgeladen. Für die durch openUTM aufzurufen-
B den Programme dieses LLMs sind zusätzlich PROGRAM-Anweisungen notwendig.

B Erzeugt ein Compiler für den shareable und den nicht-shareable Teil zwei getrennte Ob-
B jektmodule, dann sollten Sie diese Module mit dem Binder zu einem LLM mit Slices vor-
B binden. Dieses LLM können Sie dann wie oben angegeben generieren.

B Alternativ dazu können Sie auch das shareable und das nicht-shareable Modul durch
B zwei LOAD-MODULE-Anweisungen generieren. Dies sollten Sie jedoch möglichst ver-
B meiden, da sich diese beiden Module nicht ohne Konsistenzlücke austauschen lassen.

- B Einen gemeinsam nutzbaren Datenbereich (Area), der in den Common Memory Pool
B geladen werden soll, müssen Sie mit einer AREA-Anweisung beschreiben. Die Area
B muss dann in dem Lademodul enthalten sein, der folgendermaßen generiert wird:

```
B LOAD-MODULE ar-share ,VERSION=version -
B ,LOAD-MODE=(POOL,poolname,NO-PRIVATE-SLICE) -
B ,LIB=libname
```

B AREAs, denen beim Übersetzen oder durch den Binder das Attribut PUBLIC zugeord-
B net wurde, können auch zusammen mit anderen Modulen in ein LLM mit Slices vorge-
B bunden werden. Dieses LLM kann dann folgendermaßen generiert werden:

```
B LOAD-MODULE llm-with-slices ,VERSION=version -
B ,LOAD-MODE=(POOL,poolname,STARTUP)-
B ,LIB=libname
```

Beispiel

Das Beispiel geht davon aus, dass zum Übersetzen der COBOL85-Compiler verwendet wurde und dass der Compiler die Objekte in ein LLM abgelegt hat.

In den anwendungslokalen Pool LCPOOL sollen die shareable Module der COBOL-Teilprogramme TP1 und TP2 und der Datenmodul DATAMOD geladen werden. LCPOOL soll auf Adresse X'020000' geladen werden, 128 KB belegen können und schreibgeschützt sein.

```
/MPOOL         LCPOOL,SIZE=2,SCOPE=GROUP,ACCESS=READ,PAGE=X'20000'
/LOAD-MODULE   LLM-LCPOOL,VERSION=1,                              -
/              LOAD-MODE=(POOL,LCPOOL,STARTUP),                   -
/              LIB=libname
/PROGRAM       TP1     ,LOAD-MODULE=LLM-LCPOOL,COMP=ILCS
/PROGRAM       TP2     ,LOAD-MODULE=LLM-LCPOOL,COMP=ILCS
/AREA          DATAMOD,LOAD-MODULE=LLM-LCPOOL
```

Die Objektmodule müssen Sie vor dem Start der Anwendung zu dem LLM LLM-LCPOOL vorbinden und dabei in der BINDER-Anweisung START-LLM-CREATION die Option BY-ATTRIBUTES(PUBLIC=YES) angeben, wodurch das LLM in eine Public und eine Private Slice unterteilt wird. Das so erzeugte LLM müssen Sie in der Bibliothek *libname* bereitstellen.

5.7 Auftragssteuerung - Prioritäten und Prozessbeschränkung

openUTM bietet zwei Verfahren an, mit denen Sie die Verteilung der freien UTM-Prozesse auf die zur Bearbeitung anstehenden Aufträge steuern können. Das heißt, Sie können Einfluss auf die Reihenfolge nehmen, in der openUTM die Bearbeitung von Aufträgen an Transaktionscodes startet.

Durch den Einsatz eines der Verfahren zur Auftragssteuerung können Sie:

- wichtige Aufträge bevorzugt bearbeiten lassen
- verhindern, dass viele gleichartige Aufträge gleichzeitig laufen und so die Bearbeitung anderer Aufträge verzögert wird
- verhindern, dass die Auftragsbearbeitung durch so genannte Langläufer blockiert wird. Langläufer sind Services, deren Bearbeitung extrem lange dauert, z.B. weil ihre Teilprogramme Datenbestände durchsuchen oder Program Waits (blockierende Aufrufe wie z.B. PGWT) enthalten.

Bei beiden Verfahren müssen Sie die Transaktionscodes, die einer spezifischen Auftragssteuerung unterzogen werden sollen, einer TAC-Klasse zuordnen. Für die Auftragssteuerung zwischen den TAC-Klassen können Sie dann **alternativ** eines der beiden Verfahren auswählen:

- Prioritätensteuerung
  Die Verteilung der Prozesse auf die TAC-Klassen wird durch Prioritäten gesteuert, nach denen openUTM die anstehenden Aufträge bearbeiten soll. Die Prioritätensteuerung schalten Sie mit der KDCDEF-Anweisung TAC-PRIORITIES ein.
- Prozessbeschränkung
  Sie begrenzen die Anzahl der Prozesse, die gleichzeitig Aufträge einer bestimmten TAC-Klasse bearbeiten dürfen, oder Sie legen fest, wieviele Prozesse für die Bearbeitung von Aufträgen an andere TAC-Klassen frei bleiben sollen. Die Prozesszahl kann für jede TAC-Klasse einzeln festgelegt werden. Zum Festlegen der Prozessanzahl steht Ihnen die KDCDEF-Steueranweisung TACCLASS zur Verfügung.

Sie dürfen die beiden Verfahren nicht zusammen in einer Anwendung einsetzen, d.h. bei der KDCDEF-Generierung dürfen Sie die Steueranweisungen TAC-PRIORITIES und TACCLASS nicht zusammen verwenden.

Einteilung der Transaktionscodes in TAC-Klassen

openUTM unterscheidet insgesamt 16 TAC-Klassen. Für Dialog- und Asynchron-Transaktionscodes stehen jeweils 8 Klassen zur Verfügung, für Dialog-Transaktionscodes die Klassen 1 bis 8 und für Asynchron-Transaktionscodes die Klassen 9 bis 16.

Die Zuordnung der Transaktionscodes zu den TAC-Klassen legen Sie bei der KDCDEF-Generierung fest.

TAC-Anweisung auf Seite 488
Operand TACCLASS=

Für Transaktionscodes, die Sie nicht explizit einer TAC-Klasse zuordnen (keine Angabe in TACCLASS=), stellt openUTM Folgendes ein:

– Dialog-Transaktionscodes werden keiner TAC-Klasse zugeordnet
– Asynchron-Transaktionscodes werden der TAC-Klasse 16 zugeordnet

In einer TAC-Klasse sollten Sie die Transaktionscodes von gleichartigen Services zusammenfassen. Dann repräsentiert eine TAC-Klasse einen Auftragstyp Ihrer Anwendung.

Welche Aufträge unterliegen der Auftragssteuerung?

Grundsätzlich unterliegen nur die Aufträge der Auftragssteuerung, die von openUTM in eine Auftragswarteschlange eingereiht wurden.

Aufträge an Asynchron-Transaktionscodes werden immer zuerst in eine Auftragswarteschlange gestellt, bevor openUTM sie zur Bearbeitung auswählt.

Aufträge für Dialog-Transaktionscodes werden dagegen nur in Engpass-Situationen in eine Warteschlange eingereiht, z.B. wenn die Anzahl der zur Verfügung stehenden Prozesse erschöpft ist.
Ist die Auslastung der Anwendung gering, dann werden die Dialog-Aufträge sofort bearbeitet, da sie sich nicht nennenswert gegenseitig blockieren und eine Zwischenspeicherung in der Warteschlange eher verzögernd wirken würde.

Aus diesem Grund kommen die Verfahren zur Auftragssteuerung für Asynchron-Aufträge immer, für Dialog-Aufträge hingegen nur in Engpass-Situationen zur Anwendung.

Darüber hinaus unterliegen folgende Aufträge nicht der Auftragssteuerung:

- Aufträge an Dialog-Transaktionscodes, die keiner TAC-Klasse zugeordnet sind. Diese Aufträge werden immer sofort nach ihrer Entgegennahme aus dem Transportsystem gestartet.
- Aufträge an die Transaktionscodes KDCSGNTC, KDCMSGTC und KDCBADTC, über die die Event-Services (Anmelde-Vorgang, MSGTAC- und BADTACS-Programm) gestartet werden.

Verteilung der Betriebsmittel auf Dialog-, Asynchron- und PGWT-Verarbeitung steuern

In einer ersten Stufe zur Auftragsbearbeitung sollten Sie - unabhängig vom eingesetzten Verfahren zur Auftragssteuerung - festlegen, wieviele Prozesse der Anwendung maximal gleichzeitig Asynchron-Aufträge bearbeiten bzw. gleichzeitig im Program Wait warten dürfen. Auf diese Weise können Sie verhindern, dass der Dialog-Betrieb Ihrer Anwendung durch die Bearbeitung solcher Aufträge behindert wird.

MAX-Anweisung auf Seite 368
Die Prozesszahlen werden mit den folgenden Operanden generiert:

- ASYNTASKS=(*atask_number*,...)

  Mit *atask_number* legen Sie die maximale Anzahl der Prozesse der Anwendung fest, die gleichzeitig Aufträge an Asynchron-TAC-Klassen bearbeiten dürfen.

- TASKS-IN-PGWT=

  Maximale Anzahl der Prozesse der UTM-Anwendung, in denen gleichzeitig Teilprogramme mit blockierenden Aufrufen ablaufen dürfen. Sie müssen TASKS-IN-PGWT > 0 angeben, wenn Sie Transaktionscodes oder TAC-Klassen die Eigenschaft PGWT=YES zuordnen wollen.

Die in der MAX-Anweisung angegebenen Werte für ASYNTASKS=(*atask_number*,...) und TASKS-IN-PGWT sind Maximalwerte. Beim Start der Anwendung und im Anwendungsbetrieb können Sie die Prozesszahl per Administration herabsetzen, um sie der aktuellen Situation anzupassen.

Standardeinstellung

Wenn Sie keine TAC-Klassen erzeugen, d.h. in keiner TAC-Anweisung den Operanden TACCLASS angeben, dann führt openUTM keine spezielle Auftragssteuerung durch. Teilprogrammläufe mit blockierenden Aufrufen sind dann nicht erlaubt. Dialog-Aufträge werden in der Reihenfolge bearbeitet, in der sie bei openUTM eingehen.

Setzen Sie bei der Generierung weder eine TACCLASS- noch eine TAC-PRIORITIES-Anweisung ab, dann stellt openUTM automatisch das Verfahren der Prozessbegrenzung ein. Alle TAC-Klassen sind administrierbar, d.h. der UTM-Administrator kann Prozesszahlen für die TAC-Klassen festlegen.

5.7.1 Auftragsbearbeitung über Prioritäten steuern

Um die Auftragsteuerung über Prioritäten einzuschalten, müssen Sie bei der KDCDEF-Generierung die TAC-PRIORITIES-Anweisung absetzen. In ihr legen Sie auch die Algorithmen fest, nach denen die einzelnen Dialog- bzw. Asynchron-TAC-Klassen priorisiert werden sollen.

TAC-PRIORITIES-Anweisung auf Seite 512
Mit folgenden Operanden legen Sie die Algorithmen für die Prioritätensteuerung fest:

- DIAL-PRIO=
  Priorität, nach welcher die zur Verfügung stehenden Prozesse der Anwendung auf die Dialog-TAC-Klassen verteilt werden sollen.
- ASYN-PRIO=
  Priorität, nach welcher Prozesse auf die Asynchron-TAC-Klassen mit ablaufbereiten oder unterbrochenen Asynchron-Aufträgen verteilt werden sollen.

Sie können für Dialog- und Asynchron-TAC-Klassen jeweils zwischen der **absoluten**, einer **relativen** oder der **gleichen** Priorität wählen.

Unabhängig davon, welchen Algorithmus Sie wählen, gilt immer Folgendes:

– Die TAC-Klasse 1 unter den Dialog-TAC-Klassen hat eine höhere oder gleiche Priorität als die TAC-Klasse 2 und diese eine höhere oder gleiche als die TAC-Klasse 3 usw.

– Bei den Asynchron-TAC-Klassen hat die Klasse 9 eine höhere oder gleiche Priorität als die TAC-Klasse 10 und diese eine höhere oder gleiche als die TAC-Klasse 11 usw.

Bei der **absoluten Priorität** werden freie Prozesse der Anwendung immer der TAC-Klasse mit der höchsten Priorität, also 1 (Dialog) bzw. 9 (Asynchron) zugeordnet, sofern es wartende Aufträge für diese TAC-Klasse gibt. Erst wenn es keine wartenden Aufträge mehr in

der TAC-Klasse mit der höchsten Priorität gibt, werden wartende Aufträge der TAC-Klasse mit der nächst niedrigeren Priorität bearbeitet. In Zeiten hoher Auslastung können absolute Prioritäten dazu führen, dass wartende Aufträge einer TAC-Klasse mit niedriger Priorität lange Zeit nicht bearbeitet werden. Wollen Sie das verhindern, dann sollten Sie relative Prioritäten verwenden.

Bei **relativen Prioritäten** werden Aufträge aus TAC-Klassen hoher Priorität häufiger bearbeitet als Aufträge aus TAC-Klassen niedrigerer Priorität, d.h. freie Prozesse werden öfter TAC-Klassen hoher (z.B. 1) als TAC-Klassen niedriger Priorität zugeordnet, sofern für diese anstehende Aufträge vorhanden sind. Sind für alle TAC-Klassen Aufträge vorhanden, so wird Klasse 1 doppelt so oft wie Klasse 2, und diese wiederum doppelt so häufig wie Klasse 3 (und so weiter) bedient. Analoges gilt für Asynchron-TAC-Klassen.

Bei **gleichen Prioritäten** werden, sofern vorhanden, aus jeder TAC- Klasse gleich viele Aufträge bearbeitet.

Innerhalb der TAC-Klassen werden jedoch Aufträge, deren Bearbeitung zu Program Waits führt (TACs mit PGWT=YES) nur dann bearbeitet, wenn die maximal erlaubte Anzahl an Prozessen, die PGWT-Aufträge bearbeiten darf, noch nicht erreicht ist.

Prozesse für Dialog-Aufträge außerhalb der TAC-Klassen reservieren

Auch bei der Prioritätensteuerung der TAC-Klassen können Sie die Anzahl der Prozesse, die Aufträge der TAC-Klassen bearbeiten, beschränken, um Prozesse für administrative Aufgaben oder UTM-interne Aufträge frei zu halten.
Diese Beschränkung ist aber jeweils gleich für alle Asynchron- bzw. für alle Dialog-TAC-Klassen.

Die maximale Prozesszahl für Asynchron-TAC-Klassen beschränken Sie mit MAX ASYNTASKS=(*atask_number*,...), wie in Abschnitt „Verteilung der Betriebsmittel auf Dialog-, Asynchron- und PGWT-Verarbeitung steuern“ auf Seite 207 beschrieben.

Die Prozesszahl für die Dialog-TAC-Klassen beschränken Sie mit dem Operanden FREE-DIAL-TASKS= der TAC-PRIORITIES-Anweisung.

Die in FREE-DIAL-TASKS angegebene Anzahl von Prozessen wird reserviert für die Bearbeitung von Aufträgen, die keiner Dialog-TAC-Klasse angehören. Das sind Asynchron-Aufträge und Dialog-Aufträge, die keiner Dialog-TAC-Klasse zugeordnet sind, insbesondere UTM-interne Aufgaben (Aufbau von Verbindungen, Senden von Quittungen, Starten der MSGTAC-Routine usw.). Zu den UTM-internen Aufgaben gehört es auch, die für die UTM-Anwendung eingehenden Aufträge an der Auftrags-Börse abzuholen und, falls notwendig, in die Auftrags-Queues der Anwendung einzutragen. Diese „reservierten Prozesse“ bewirken somit, dass die Börse entlastet wird. Insbesondere wenn viele Aufträge an die Anwendung aus dem Netz kommen, verhindert das einen Rückstau ins Netz bis hin zum Kommunikationspartner.

Wieviele Prozesse Sie für diese Aufgaben reservieren sollten, ist abhängig von Ihrer Anwendung. Es wird empfohlen einen oder zwei Prozesse für diese Aufgaben freizuhalten.

Die Anzahl der freien Prozesse können Sie per Administration ändern.

Siehe openUTM-Handbuch „Anwendungen administrieren“; KDCADMI-Operationscode KC_MODIFY_OBJECT mit Objekttyp KC_TASKS_PAR

Beispiel

Bei der KDCDEF-Generierung werden folgende maximale Prozesszahlen festgelegt:

```
MAX TASKS=7,ASYNTASKS=2
TAC-PRIORITIES ...,FREE-DIAL-TASKS=3
```

Wird die Anwendung dann mit sechs Prozessen gestartet (Startparameter TASKS=6), dann stehen folgende Prozesszahlen zur Verfügung:

– Drei Prozesse für die Bearbeitung von Aufträgen an die Dialog-TAC-Klassen 1 bis 8. (Bestimmt durch: TASKS – FREE-DIAL-TASKS = 6 – 3 = 3)

– Zwei (=ASYNTASKS) Prozesse für die Bearbeitung von Aufträgen an die Asynchron-TAC-Klassen 9 bis 16.

– Ein Prozess für UTM-interne Aufgaben und Dialog-Aufträge an Transaktionscodes, die keiner TAC-Klasse zugeordnet sind.
(Bestimmt durch: FREE-DIAL-TASKS – ASYNTASKS = 3 – 2 = 1

5.7.2 Auftragsbearbeitung durch Prozessbeschränkung für TAC-Klassen steuern

Die Auftragssteuerung über Prozessbeschränkung wird über die TACCLASS-Anweisung generiert. Die Prozessbeschränkung erfolgt TAC-Klassen-spezifisch, d.h. Sie können für jede TAC-Klasse eine eigene TACCLASS-Anweisung absetzen.

TACCLASS-Anweisung auf Seite 506
Zur Einstellung der Prozessbegrenzung können Sie alternativ einen der beiden folgenden Operanden angeben:

- TASKS=

  Anzahl der Prozesse, die maximal Aufträge für diese TAC-Klasse bearbeiten dürfen.

- TASKS-FREE=

  Anzahl der Prozesse, die mindestens für die Bearbeitung von Aufträgen anderer TAC-Klassen bzw. von Aufträgen, die keiner TAC-Klasse zugeordnet sind, freigehalten werden soll.

Bei diesem Verfahren sagen die Nummern der TAC-Klassen nichts über die Priorität aus, mit der ihre Aufträge bearbeitet werden. Nur die Zahl der Prozesse, die Sie für diese TAC-Klasse zulassen, gibt an, wie stark die Bearbeitung der Aufträge gegenüber anderen TAC-Klassen gedrosselt wird.

Dieses Verfahren kann dann sinnvoll eingesetzt werden, wenn in einer Anwendung nur wenige unterschiedliche Auftragstypen - und damit wenige TAC-Klassen - vorkommen, und z.B. verhindert werden soll, dass langlaufende Aufträge alle Prozesse einer Anwendung belegen und damit die Bearbeitung von anderen, wichtigen Aufträgen - wie z.B. Administrationsaufträge - unnötig behindern.

5.7.3 Gegenüberstellung einiger Eigenschaften der beiden Verfahren

In Ihrer UTM-Anwendung können Sie nur eines der beiden Verfahren zur Auftragssteuerung einsetzen. Welche der beiden Möglichkeiten Sie für Ihre Anwendung wählen sollten, ist auch von den z.T. unterschiedlichen Eigenschaften der beiden Verfahren abhängig, die im Folgenden gegenübergestellt werden.

Teilprogramme mit blockierenden Aufrufen

- *Prioritätensteuerung*

  Transaktionscodes von Teilprogrammen, die blockierende Aufrufe durchlaufen, dürfen jeder beliebigen TAC-Klasse zugeordnet werden, sofern im Operanden TASKS-IN-PGWT der MAX-Anweisung ein Wert > 0 generiert ist. Für Transaktionscodes mit blockierenden Aufrufen müssen Sie in der TAC-Anweisung den Operanden PGWT=YES angeben.

  ```
  TAC ..., TACCLASS=number,PGWT=YES
  ```

  Damit können auch entsprechende Aufträge mit verschiedenen Prioritäten bearbeitet werden.

- *Prozessbeschränkung*

  Alle Transaktionscodes von Teilprogrammen, die blockierende Aufrufe durchlaufen, müssen derselben Dialog- bzw. Asynchron-TAC-Klasse zugeordnet werden. Diese Dialog- bzw. Asynchron-TAC-Klasse müssen Sie wie folgt generieren:

  ```
  TACCLASS ...,PGWT=YES
  ```

  Die entsprechenden Dialog- bzw. Asynchron-Aufträge werden also gleich behandelt.

Ausführung bestimmter Asynchron-Aufträge vorübergehend stoppen

Beide Verfahren zur Auftragssteuerung bieten einen Mechanismus an, mit dem Sie die Bearbeitung bestimmter Asynchron-Aufträge vorübergehend verhindern können. Diese Aufträge werden dann zwar von openUTM entgegengenommen und in die Message Queue des entsprechenden Transaktionscodes geschrieben. Die Bearbeitung dieser Aufträge erfolgt jedoch erst, wenn „die Bearbeitungssperre“ durch die UTM-Administration aufgehoben wird.

Um die Durchführung von Aufträgen vorübergehend zu verhindern, setzen Sie den Status des Transaktionscodes auf KEEP. Das können Sie im laufenden Betrieb über die UTM-Administration tun oder bereits bei der Generierung des Transaktionscodes, indem Sie Folgendes angeben:

```
TAC ...,STATUS=KEEP
```

openUTM bearbeitet die zwischengespeicherten Aufträge erst, wenn Sie den Status des Transaktionscodes auf ON setzen.

Siehe openUTM-Handbuch „Anwendungen administrieren“; KDCADMI-Operationscode KC_MODIFY_OBJECT mit Objekttyp KC_TAC oder Administrationskommando KDCTAC

Beim Verfahren der *Prozessbeschränkung* kann die Durchführung von Aufträgen zusätzlich für alle Transaktionscodes einer Asynchron-TAC-Klasse verhindert werden. In diesem Fall müssen Sie die maximale Anzahl der Prozesse, die für Aufträge dieser TAC-Klasse zur Verfügung stehen, auf 0 herabsetzen.

```
TACCLASS ...,TASKS=0
```

openUTM bearbeitet die Aufträge erst, wenn Sie die Prozesszahl wieder heraufsetzen.

Siehe openUTM-Handbuch „Anwendungen administrieren“; KDCADMI-Operationscode KC_MODIFY_OBJECT mit Objekttyp KC_TACCLASS oder Administrationskommando KDCTCL

Beide Mechanismen können Sie z.B. dazu verwenden, um Aufträge zu sammeln, die erst zu einem Zeitpunkt ausgeführt werden sollen, an dem die Anwendung weniger belastet ist (z.B. nachts).

In beiden Fällen sollten Sie, um eine Überlastung des Pagepools durch zuviele zwischengespeicherte Aufträge zu vermeiden, die Message Queue des/der Transaktionscodes begrenzen. Dies geschieht TAC-spezifisch durch:

```
TAC ...,QLEV=
```

Prozesswechsel bei der Bearbeitung von Aufträgen

- *Prioritätensteuerung*

  Besteht ein Vorgang aus mehreren Teilprogrammen (Folge-TAC nach PEND PA/PR), dann kann es bei der Bearbeitung des Vorgangs immer zu einem Prozesswechsel kommen - unabhängig davon, ob aktueller TAC und Folge-TAC zur selben TAC-Klasse gehören oder nicht.

- *Prozessbeschränkung*

  Bei der Auftragssteuerung über Prozessbeschränkung garantiert openUTM, dass nach einem PEND PA/PR und SP kein Prozesswechsel stattfindet, wenn Vorgangs- und Folge-TAC derselben TAC-Klasse zugeordnet sind.

  Gehören aktueller TAC und Folge-TAC zu verschiedenen TAC-Klassen, kann es auch bei diesem Verfahren zum Prozesswechsel kommen.

Prozesswechsel bei Asynchron-Vorgängen

Bei einem Prozesswechsel wird ein Asynchron-Vorgang zunächst inaktiv und belegt keinen UTM-Prozess, bleibt aber offen.

Die maximale Anzahl gleichzeitig offener Asynchron-Vorgänge können Sie beschränken. Dazu müssen Sie in der MAX-Anweisung Folgendes angeben:

MAX ...,ASYNTASKS=(...,*service_number*).

Sind *service_number* offene Asynchron-Vorgänge vorhanden, wird kein neuer anstehender Asynchron-Auftrag mehr gestartet, sondern vom nächsten freiwerdenden Prozess ein unterbrochener offener Asynchron-Vorgang ausgewählt und fortgesetzt.

5.7.4 Prozessprioritäten in BS2000/OSD

openUTM verwendet die oben beschriebenen Verfahren zur Auftragssteuerung, um einen Auftrag auszuwählen, der als nächstes neu gestartet bzw. fortgesetzt werden soll. Aufträge, die aktuell bearbeitet werden, können mit diesen Verfahren nicht beeinflusst werden.

Um auch auf die Priorität der aktiven Aufträge Einfluss zu nehmen, können Sie die Scheduling Mechanismen des BS2000/OSD für die Priorisierung von Aufträgen einsetzen. Dazu dient der Operand RUNPRIO der TAC-Anweisung. Mit RUNPRIO weisen Sie einem Transaktionscode bei der KDCDEF-Generierung eine Prozesspriorität (Run-Priorität) zu. Mit der Prozesspriorität können Sie die Geschwindigkeit beeinflussen, mit der ein laufender Auftrag bearbeitet wird. Ein Auftrag an einen Transaktionscode mit höherer Prozesspriorität wird gegenüber anderen Aufträgen niedrigerer Priorität bei der CPU-Zuteilung bevorzugt.

Haben Sie für einen Transaktionscode eine Prozesspriorität generiert, dann setzt openUTM die BS2000-Prozesspriorität des Prozesses, der einen Auftrag für diesen Transaktionscode bearbeitet, auf den in RUNPRIO generierten Wert.
In RUNPRIO können Sie einen Wert zwischen 30 (höchste Priorität) und 255 (niedrigste Priorität) angeben.

TAC-Anweisung auf Seite 488
Operand RUNPRIO

5.8 Zugriffskontrolle

Bei Services, die auf sicherheitsrelevante Daten zugreifen, ist es sinnvoll, den Zugriff auf wenige, autorisierte Benutzer zu beschränken. openUTM bietet zwei alternative Zugriffskontrollverfahren, die es ermöglichen, die Zugriffsrechte in einer UTM-Anwendung differenziert festzulegen:

– Access-List-Konzept (Service-orientiert)

– Lock-/Keycode-Konzept (Benutzer-orientiert)

Beide Verfahren verwenden weitgehend die gleichen Generierungsschnittstellen.
Der wesentliche Unterschied liegt in der Sicht auf die UTM-Objekte: Während man mit dem Access-List-Konzept für jeden einzelnen Service mittels einer Liste von Codes festlegt, welche Benutzer(-typen) auf ihn zugreifen dürfen, definiert man im Lock-/Keycode-Konzept für jeden Service einen (einzelnen) Lockcode und weist dann jedem Benutzer die entsprechenden Keycodes zu.

Services, deren TACs nicht durch einen Lockcode oder eine Access-List gesichert sind, kann jeder Benutzer ohne Einschränkung aufrufen.

Für ausführliche Informationen zum Access-List- und zum Lock-/Keycode-Konzept siehe openUTM-Handbuch „Konzepte und Funktionen“.

5.8.1 Lock-/Keycode-Konzept

Ein Lockcode ist eine Zahl, die ein logisches Schloss darstellt. Die zu schützenden Objekte - z.B. LTERM-Partner und den Services zugeordnete Transaktionscodes - werden mit einem solchen Lockcode versehen (TAC- bzw. LTERM-Anweisung).
Für Benutzerkennungen und LTERM-Partner werden Keycodes definiert (USER- bzw. LTERM-Anweisung). Nur wenn ein Keycode mit dem Lockcode eines gesicherten Objekts übereinstimmt, ist der Zugriff auf dieses Objekt erlaubt.
Da in der Regel eine Benutzerkennung oder ein LTERM-Partner Zugriff auf mehrere Services hat, verfügen sie über mehrere Keycodes. Die einzelnen Keycodes werden daher jeweils zu Keysets zusammengefasst (KSET-Anweisung).

Das Lock-/Keycode-Konzept hat folgende Effekte:

– Die Anmeldung eines (UTM-)Benutzers / unter einer UTM-Benutzerkennung ist nur möglich, wenn der angegebenen Benutzerkennung ein Keycode zugeordnet ist, der mit dem Lockcode des LTERM-Partners, über den die Anmeldung erfolgt, übereinstimmt.

– Ein Benutzer kann einen Service nur dann aufrufen, wenn **sowohl** das Keyset der jeweiligen (UTM-)Benutzerkennung **als auch** das des LTERM-Partners einen Keycode enthalten, der mit dem Lockcode des Transaktionscodes übereinstimmt.

KSET-Anweisung auf Seite 334
Mit den folgenden Operanden können Keysets definiert werden:

- *keysetname*

  Name des Keysets.

- KEYS=

  bei Vergabe eines Keysets an einen Benutzer (USER):
  Angabe eines oder mehrerer Keycodes (numerisch), die dem Benutzer zugeordnet werden.

  bei Vergabe eines Keysets an einen LTERM-Partner (LTERM):
  Angabe eines oder mehrerer Keycodes (numerisch), die dem LTERM-Partner zugeordnet werden.

TAC-Anweisung auf Seite 488
Mit den folgenden Operanden werden die Zugriffe auf den TAC kontrolliert:

- *tacname*

  Name des TACs.

- LOCK=

  gibt den Lockcode an, der dem TAC eines Services als logisches Zahlenschloss zugeordnet ist.

  Ein mit Lockcode gesicherter Service kann dann nur gestartet werden, wenn im Keyset des Benutzers (USER) **und** im Keyset des LTERM-Partners ein mit dem Lockcode übereinstimmender Keycode enthalten ist.

  Darf nicht zusammen mit dem Operanden ACCESS-LIST= angegeben werden.

USER-Anweisung auf Seite 538
Mit den folgenden Operanden wird einem Benutzer ein Keyset zugeordnet:

- *username*

  UTM-Benutzerkennung.

- KSET=

  Gibt den Namen des Keysets an, das der Benutzerkennung zugeordnet wird. Das Keyset muss mit der KSET-Anweisung definiert worden sein. Jedem Benutzer kann max. ein Keyset zugeordnet werden.
  Ein Benutzer kann auf einen Service, dessen erster TAC durch einen Lockcode geschützt ist, nur dann zugreifen, wenn einer der Keycodes in seinem Keyset mit dem Lockcode übereinstimmt. Sonst wird der Zugriff auf den Service verweigert.

LTERM-Anweisung auf Seite 354 / **TPOOL-Anweisung** auf Seite 518
Mit den folgenden Operanden wird einem LTERM-Partner ein Keyset zugeordnet:

- *ltermname*

  Name des LTERM-Partners (nur für LTERM-Anweisung).

- LTERM= , NUMBER=

  Name des LTERM-Partners (nur für TPOOL-Anweisung).

- KSET=

  Gibt den Namen des Keysets an, das dem LTERM-Partner zugeordnet wird. Das Keyset muss mit der KSET-Anweisung definiert worden sein. Bei LTERM-Partnern eines UPIC-Clients oder einer TS-Anwendung ohne explizit generierte Verbindungs-Benutzerkennung ist dieses Keyset zugleich das Keyset der Verbindungs-Benutzerkennung.

- USER-KSET= (nur für TPOOL-Anweisung)

  Gibt in LTERM-Pools für TS-Anwendungen oder UPIC-Clients den Namen des Keysets an, das der Verbindungs-Benutzerkennung zugeordnet wird. Das Keyset muss mit der KSET-Anweisung definiert worden sein. Die Zugriffsrechte ergeben sich aus der Schnittmenge der Keysets von KSET= und USER-KSET=.

- LOCK=

  Lockcode, der dem LTERM-Partner als logisches Zahlenschloss zugeordnet ist. Gilt nur für Clients (USAGE=D).

  Über einen zugriffsgesicherten LTERM-Partner kann sich nur ein (UTM-)Benutzer an die Anwendung anmelden, für den ein Keyset mit einem Keycode generiert wurde, der mit dem Lockcode des LTERM-Partners übereinstimmt.

5.8.2 Access-List-Konzept

Eine Access-List ist eine Menge von Zugangscodes (numerische Codes), die einem Service zugeordnet wird. Die Zugangscodes in der Access-List können als Rollen der Benutzer innerhalb ihrer Organisationsstruktur aufgefasst werden (z.B. Sachbearbeiter, Abteilungsleiter, Administrator), die Zugriff auf diesen Service haben sollen.
Wenn Sie das Administrationstool WinAdmin verwenden, können Sie statt der numerischen Codes auch sprechende Namen verwenden.

Eine Access-List wird mit der KSET-Anweisung definiert und mit der TAC-Anweisung einem Service zugeordnet. Die Rollen für den Benutzer (USER) werden ebenfalls mit einer KSET-Anweisung als Keyset definiert und zugewiesen. In gleicher Weise kann einem LTERM-Partner eine bestimmte Menge von Rollen zugeordnet werden.

Ein Benutzer kann nur dann auf einen so geschützten Service (TAC) zugreifen, wenn sowohl das Keyset des Benutzers als auch das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, jeweils mindestens eine der Rollen enthält, die in der Access-List des Service enthalten sind.

Die Unterschiede von Lock-/Keycode- und Access-List-Konzept sind ausführlich im openUTM-Handbuch „Konzepte und Funktionen“, Security-Funktionen, beschrieben.

KSET-Anweisung auf Seite 334
Mit den folgenden Operanden können Keysets oder Access-Lists definiert werden:

- *keysetname*

  Name des Keysets bzw. der Access-List.

- KEYS=

  bei Vergabe einer Access-List an einen Service (TAC):
  Angabe einer oder mehrerer Rollen (als numerische Werte), die Zugriff auf den mit *keysetname* geschützten Service haben.

  bei Vergabe eines Keysets an einen Benutzer (USER):
  Angabe einer oder mehrerer Rollen (als numerische Werte), die dem Benutzer zugeordnet werden sollen.

  bei Vergabe eines Keysets an einen LTERM-Partner (LTERM):
  Angabe einer oder mehrerer Rollen (als numerische Werte), die bei Anmeldung über diesen LTERM-Partner ausgeübt werden dürfen.

Beim Einsatz von WinAdmin können die Rollen auch mit alphanumerischen Namen vergeben werden.

TAC-Anweisung auf Seite 488
Mit den folgenden Operanden werden die Zugriffe auf den TAC kontrolliert:

- *tacname*

  Name des TACs.

- ACCESS-LIST=

  Gibt die Access-List an, die den Zugriff auf diesen TAC kontrolliert. Nur Benutzern, deren Keyset mindestens eine der Rollen in dieser Access-List enthält und die sich über ein Terminal anmelden, das auch mindestens eine der Rollen zugeordnet bekommen hat, ist der Zugriff auf diesen TAC erlaubt.
  ACCESS-LIST darf nicht zusammen mit LOCK angegeben werden.

USER-Anweisung auf Seite 538
Mit den folgenden Operanden wird einem Benutzer ein Keyset zugeordnet:

- *username*

  UTM-Benutzerkennung.

- KSET=

  Gibt den Namen des Keysets an, das der Benutzerkennung zugeordnet wird. Das Keyset muss mit der KSET-Anweisung definiert worden sein. Jedem Benutzer kann max. ein Keyset zugeordnet werden.
  Will ein Benutzer auf einen Service zugreifen, der mit einer Access-List geschützt ist, muss mindestens eine seiner Rollen in der Access-List vorhanden sein. Sonst wird der Zugriff auf den Service verweigert.

LTERM-Anweisung auf Seite 354 / **TPOOL-Anweisung** auf Seite 518
Mit den folgenden Operanden wird einem LTERM-Partner ein Keyset zugeordnet:

- *ltermname*

  Name des LTERM-Partners (nur für LTERM-Anweisung).

- LTERM= , NUMBER=

  Name des LTERM-Partners (nur für TPOOL-Anweisung).

- KSET=

  Gibt den Namen des Keysets an, das dem LTERM-Partner zugeordnet wird. Bei LTERM-Partnern eines UPIC-Clients oder einer TS-Anwendung ohne explizit generierte Verbindungs-Benutzerkennung ist dieses Keyset zugleich das Keyset der Verbindungs-Benutzerkennung. Das Keyset muss mit der KSET-Anweisung definiert worden sein. Jedem LTERM-Partner kann max. ein Keyset zugeordnet werden.

- USER-KSET= (nur für TPOOL-Anweisung)

  Gibt in LTERM-Pools für TS-Anwendungen oder UPIC-Clients den Namen des Keysets an, das der Verbindungs-Benutzerkennung zugeordnet wird. Das Keyset muss mit der KSET-Anweisung definiert worden sein. Die Zugriffsrechte ergeben sich aus der Schnittmenge der Keysets von KSET= und USER-KSET=.

Der Zugriff auf LTERM-Partner kann nicht mit Access-Lists geschützt werden. Bei der Nutzung von Access-Lists zur Zugriffskontrolle auf Services sollte deshalb auf einen Zugriffsschutz auf LTERM-Partner verzichtet, d.h. der Parameter LOCK der LTERM- und TPOOL-Anweisung nicht angegeben werden.

Zugriffsschutz für Service-gesteuerte Queues mit Access-Lists

Auch Service-gesteuerte Queues können vor unbefugtem Lesen und Löschen oder Schreiben geschützt werden. Dazu wird eine Access-List definiert (TAC-/USER-Anweisung).

TAC-Anweisung auf Seite 488
Mit den folgenden Operanden können Sie den Zugriff für TAC-Queues kontrollieren:

- *tacname*

  Name der TAC-Queue.

- Q-READ-ACL=
- Q-WRITE-ACL=

  Name der Access-List, die den Lese- und Lösch- bzw. Schreibzugriff eines Benutzers auf diese Queue kontrolliert. Die Access-List muss mit der KSET-Anweisung generiert worden sein.

Ein Benutzer kann nur dann lesend oder schreibend auf die TAC-Queue zugreifen, wenn das Keyset des Benutzers und das Keyset des LTERM-Partners, über das der Benutzer angemeldet ist, jeweils mindestens eine Rolle der für die TAC-Queue definierten Access-List enthalten.
Das Keyset muss für den Benutzer und den LTERM-Partner mit USER- bzw. LTERM-Anweisungen generiert werden.

USER-Anweisung auf Seite 538
Mit den folgenden Operanden können Sie den Zugriff für USER-Queues kontrollieren:

- *username*

  UTM-Benutzerkennung.

- KSET=

  Gibt den Namen des Keysets an, das der Benutzerkennung zugeordnet wird. Das Keyset muss mit der KSET-Anweisung definiert worden sein. Jedem Benutzer kann max. ein Keyset zugeordnet werden.

- Q-READ-ACL=
- Q-WRITE-ACL=

  Name der Access-List, über die der Benutzer seine eigenen USER-Queues vor Lesen und Löschen oder Schreiben schützt. Die Access-List muss mit der KSET-Anweisung generiert worden sein.

  Der Eigentümer einer Queue hat immer Lese-, Lösch- und Schreibrechte in seiner USER-Queue, unabhängig davon, ob der Lese- bzw. Schreibzugriff für andere Benutzer beschränkt ist.

  Ein fremder Benutzer kann nur dann lesend oder schreibend auf die USER-Queue eines anderen Benutzers zugreifen, wenn das Keyset des fremden Benutzers und das Keyset des LTERM-Partners, über das der fremde Benutzer angemeldet ist, jeweils mindestens eine Rolle der für die USER-Queue definierten Access-List enthalten.

  Geben Sie Q-READ-ACL/Q-WRITE-ACL nicht an, dann hat jeder Benutzer Lese- und Lösch- bzw. Schreibrechte in der Queue.

Für ausführliche Informationen zu Message Queues siehe Seite 178ff.

5.8.3 Zugriffskontrolle bei verteilter Verarbeitung

Auch bei verteilter Verarbeitung können Sie die Mechanismen von openUTM zur Zugriffskontrolle nutzen. Die Schutzmaßnahmen werden bei der Generierung der Anwendungen festgelegt.

Schutzmaßnahmen in der Auftraggeber-Anwendung

Bei der Generierung einer Anwendung legen Sie zunächst generell fest, welche Services einer fernen Partner-Anwendung aufrufbar sein sollen: Für jeden fernen Service, der genutzt werden soll, vereinbaren Sie einen lokalen Transaktionscode LTAC (LTAC-Anweisung). Der Zugriff auf ferne Services, für die keine LTACs vereinbart sind, bleibt der Anwendung grundsätzlich verwehrt.

Um den Zugriffsschutz weiter zu differenzieren, können Sie die einzelnen LTACs mit Lockcodes (siehe Seite 216) oder mit Access-Lists (siehe Seite 219) versehen. Ein Service der lokalen Anwendung kann einen fernen Service nur dann adressieren, wenn der Service unter einer Benutzerkennung (KCBENID) und von einem Client (KCLOGTER) aus gestartet wurde, die die entsprechenden Zugriffsrechte haben.

LTAC-Anweisung auf Seite 347
Mit den folgenden Operanden wird festgelegt, welcher Service einer fernen Partner-Anwendung aufrufbar sein soll, und welche Zugriffsrechte auf den LTAC bestehen. Die Operanden ACCESS-LIST und LOCK schließen sich aus.

- *ltacname*

  Name eines lokalen TACs (LTAC) für das ferne Service-Programm.

- ACCESS-LIST=

  Name einer Access-List. Um das ferne Service-Programm starten zu dürfen, muss ein Benutzer der lokalen UTM-Anwendung in seinem Keyset mindestens eine zur Access-List passende Rolle zugewiesen bekommen haben (definiert in der USER-Anweisung).
  Die Access-List muss mit einer KSET-Anweisung definiert werden.

- LOCK=

  Definition des Lockcodes des fernen Service-Programms. Ein Service der lokalen Anwendung kann diesen fernen Service nur dann adressieren, wenn der lokale Service unter einer Benutzerkennung (KCBENID) und von einem Client (KCLOGTER) aus gestartet wurde, die die entsprechenden Zugriffsrechte haben.
  ACCESS-LIST und LOCK können nicht gleichzeitig angegeben werden.

Geben Sie weder ACCESS-LIST noch LOCK an, dann ist der LTAC nicht geschützt und jeder Benutzer der lokalen UTM-Anwendung kann das ferne Service-Programm adressieren.

Schutzmaßnahmen in der Auftragnehmer-Anwendung

In der Auftragnehmer-Anwendung (Partner-Server-Anwendung) wird der Auftraggeber-Anwendung ein Keyset zugeordnet. Nur wenn dieses Keyset einen Keycode bzw. einen Zugangscode enthält, der zum Lockcode bzw. zur Access-List des angeforderten Services passt, wird der von der Auftraggeber-Anwendung angeforderte Vorgang gestartet.

Damit auf einen fernen Service zugegriffen werden kann, muss der gerufene Service mit einem TAC generiert sein und es müssen folgende Bedingungen erfüllt sein:

- LU6.1-Verbindungen:

  Das in LPAP ...,KSET= definierte Keyset des Partners muss einen Keycode enthalten, der zu LOCK= bzw. ACCESS-LIST= des TAC passt.

- OSI TP-Verbindungen:

  – Meldet sich der Partner ohne Benutzerkennung an, dann muss sowohl das in OSI-LPAP ... ,KSET= als auch das in OSI-LPAP ...,ASS-KSET= definierte Keyset einen Schlüssel enthalten, der zu LOCK= bzw. ACCESS-LIST= des TAC passt.

    Die Zugriffsrechte ergeben sich aus der Schnittmenge der Keysets von KSET= und ASS-KSET=. Daher sollte KSET= immer eine Obermenge von ASS-KSET= sein. Durch geeeignete Beschränkung des mit OSI-LPAP ...,ASS-KSET= definierten Keyset können Sie erreichen, dass bestimmte TACs nur aufgerufen werden können, wenn der Partner eine echte Benutzerkennung angibt.

  – Meldet sich der Partner mit einer echten Benutzerkennung an, dann müssen das Keyset dieser Benutzerkennung und das in OSI-LPAP ...,KSET= definierte Keyset einen Schlüssel enthalten, der zu LOCK= bzw. ACCESS-LIST= des TAC passt.

  Dies gilt auch für eine Client-Server-Kopplung mit OpenCPIC.

Für ausführliche Informationen zur Zugriffskontrolle bei verteilter Verarbeitung siehe openUTM-Handbuch „Konzepte und Funktionen“.

5.9 Nachrichten-Verschlüsselung auf Verbindungen zu Clients

Clients greifen häufig über offene Netze auf UTM-Services zu. Damit besteht die Möglichkeit, dass Unbefugte auf der Leitung mitlesen und z.B. Passwörter für UTM-Benutzerkennungen oder sensible Benutzerdaten ermitteln. Um dies zu verhindern, unterstützt openUTM die Verschlüsselung von Passwörtern und Benutzerdaten auf Verbindungen zu UPIC-Clients und unter BS2000/OSD zusätzlich auf Verbindungen zu bestimmten Terminal-Emulationen. Für Verbindungen zu UPIC-Clients ab V5.2 sind mehrere Verschlüsselungsebenen wählbar (DES-Schlüssel oder AES-Schlüssel mit drei unterschiedlichen Schlüssellängen, siehe Seite 229).

Die Verschlüsselung dient bei openUTM nicht nur der Datensicherung auf der Verbindung zwischen Client und Server-Anwendung, sondern kann auch dazu verwendet werden, den Zugang durch Clients und den Zugriff auf bestimmte Services einzuschränken. Dabei lassen sich zwei Verschlüsselungsebenen auswählen (DES- oder AES-Verfahren, siehe Seite 229).

5.9.1 Voraussetzungen

Verbindung Server-Anwendung - UPIC-Client

Voraussetzung für die Verschlüsselung zwischen einer UTM-Server-Anwendung und einem UPIC-Client ist:

- Für die Verschlüsselung auf Verbindungen zu UPIC-Clients muss auf Server-Seite im UTM-Systemcode die Verschlüsselungskomponente openUTM-Crypt verfügbar sein. Aus rechtlichen Gründen werden die Verschlüsselungs-Funktionen von openUTM als eigenes Produkt openUTM-CRYPT ausgeliefert, das separat installiert werden muss. Siehe dazu die Freigabemitteilung zu openUTM und für openUTM auf BS2000 auch die Installationsinformationen im openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“.
- Auf UPIC-Client-Seite muss openUTM-Client für Trägersystem UPIC ab V5.0 mit Verschlüsselungs-Funktionen im Einsatz sein.

B **Verbindung Server-Anwendung - Terminal-Emulation (BS2000/OSD)**

B Für Verbindungen einer UTM-Anwendung unter BS2000/OSD zu Terminal-Emulationen
B wird die Verschlüsselung von VTSU angeboten. VTSU-B verwendet eine eigene Schlüssel-
B verwaltung. Somit kommt auf Verbindungen zu einer Terminal-Emulation die Verschlüsse-
B lung von openUTM nicht zum Einsatz, das Produkt openUTM-CRYPT ist nicht erforderlich.
B Die mit der Verschlüsselung verbundenen Zugriffs- und Zugangsschutzmechanismen sind
B jedoch wirksam. openUTM erhält von VTSU Informationen über die Verschlüsselungsebe-
B ne, die für die Verbindung zum Client vereinbart ist.

B Folgende Voraussetzungen müssen erfüllt sein:

B
B
B
B
- Voraussetzung ist der Einsatz von VTSU-B und der Liefereinheit VTSU-SEC. Welche aktuellen Versionen Sie verwenden müssen, entnehmen Sie der Freigabemitteilung zu openUTM. Welche VTSU-Parameter gesetzt werden müssen, können Sie der Freigabemitteilung zu VTSU-SEC entnehmen.

B
B
- Auf Client-Seite muss eine Terminal-Emulation zum Einsatz kommen, die die Verschlüsselungs-Funktionen unterstützt (z.B. DESK2000).

B Diese Kommunikationspartner werden im Folgenden VTSU-Partner genannt.

5.9.2 Verschlüsselungs-Verfahren

openUTM verwendet zum Verschlüsseln eine Kombination aus AES-Verfahren (Advanced Encryption Standard) und RSA-Verfahren (benannt nach den Autoren Rivest, Shamir und Adleman). Für Partner, die noch nicht das AES-Verfahren unterstützen, kann auch noch das DES-Verfahren (Data Encryption Standard) verwendet werden.

AES- und DES-Verfahren

Benutzerdaten und Passwörter auf einer Verbindung werden mit einem symmetrischen AES- oder DES-Schlüssel verschlüsselt, d.h. Client und UTM-Anwendung verwenden denselben AES-/DES-Schlüssel beim Verschlüsseln und Entschlüsseln der Nachrichten.

Der AES-/DES-Schlüssel wird vom Client erzeugt und bei Verbindungsaufbau an die UTM-Anwendung übertragen. Der Schlüssel ist Verbindungs-spezifisch, d.h. es wird für jede Verbindung ein eigener Schlüssel erzeugt und dieser wird nur für diese eine Verbindung verwendet.

RSA-Verfahren

Zur Erhöhung der Sicherheit wird der AES- bzw. DES-Schlüssel selbst vor der Übertragung verschlüsselt. Dazu erzeugt openUTM bei der Generierung ein RSA-Schlüsselpaar, das aus einem öffentlichen und einem geheimen privaten Schlüssel besteht. Das RSA-Schlüsselpaar ist Anwendungs-spezifisch und dient dazu, den AES-/DES-Schlüssel auf allen entsprechenden Verbindungen der UTM-Anwendung zu Clients zu verschlüsseln:

- Der öffentliche RSA-Schlüssel wird direkt beim Aufbau der Verbindung von der UTM-Anwendung an den Client übertragen.
- Der Client verschlüsselt damit den AES-/DES-Schlüssel und überträgt ihn an die UTM-Anwendung.
- Die UTM-Anwendung entschlüsselt den AES-/DES-Schlüssel mit Hilfe des zugehörigen privaten RSA-Schlüssels.

Für die Verschlüsselung sind mehrere Verschlüsselungsebenen wählbar. Sie unterscheiden sich in der Länge der verwendeten RSA-Schlüssel. Daher können in einer UTM-Anwendung gleichzeitig mehrere RSA-Schlüsselpaare mit unterschiedlichen Schlüssellängen existieren.

Der Administrator der Anwendung kann jederzeit neue RSA-Schlüsselpaare erzeugen und aktivieren (siehe Abschnitt „RSA-Schlüsselpaar erzeugen und öffentlichen Schlüssel auslesen“ auf Seite 231).

RSA-Schlüsselpaare werden von KDCUPD aus einer alten KDCFILE in die neue KDCFILE übernommen.

B **Verschlüsselungsverfahren bei BS2000-Terminalemulationen**

B Das RSA-Schlüsselpaar der UTM-Anwendung wird für Verbindungen zu VTSU-Partnern
B nicht verwendet. Hier kommt ein von VTSU-B erzeugtes Schlüsselpaar zum Einsatz.
B VTSU-B verwendet für die Verschlüsselung die gleichen Algorithmen und Verfahren wie
B openUTM.

5.9.3 Verschlüsselung von Passwörtern und Benutzerdaten

Auf Verbindungen zwischen UTM-Anwendung und vertrauenswürdigen Clients (als trusted generiert; siehe Punkt 3 auf Seite 229) werden weder Benutzerdaten noch Passwörter verschlüsselt übertragen.

In openUTM werden **Passwörter** von (not-trusted) UPIC-Clients immer verschlüsselt an die UTM-Anwendung übertragen, wenn sowohl Client als auch Server Verschlüsselung unterstützen. Passwörter werden in diesem Fall auch dann verschlüsselt, wenn für die Verbindung keine Verschlüsselung vereinbart wurde.

B Auf Verbindungen zwischen einer UTM-Anwendung unter BS2000/OSD und VTSU-Part-
B nern werden Passwörter nur dann verschlüsselt übertragen, wenn auf der Verbindung Ver-
B schlüsselung vereinbart wurde oder wenn das Passwort in ein dunkelgesteuertes Feld ein-
B gegeben wurde.

Die Verschlüsselung der **Benutzerdaten** ist optional. Sie wird beim Aufbau einer UPIC-Conversation bzw. Verbindung zu einem VTSU-Partner zwischen Client und Server ausgehandelt.

– Der Client kann Verschlüsselung erzwingen.
  Einem UPIC-Client stehen dazu das Schlüsselwort ENCRYPTION-LEVEL in der Side Information-Datei und der Funktions-Aufruf *Set_Encryption_Level* zur Verfügung.
  B Für VTSU-Partner wird die Verschlüsselungsebene am Host eingestellt. Es können von
  B unbedingter Verschlüsselung für alle Anwendungen bis zur Verschlüsselung einzelner
  B Nachrichten, die der Benutzer selbst auswählt, verschiedene Verschlüsselungsebenen
  B eingestellt werden.

– Eine UTM-Anwendung kann für einen bestimmten Service oder einen bestimmten Partner Verschlüsselung anfordern.

Fordert einer der Partner Verschlüsselung an, dann wird entweder die Verschlüsselung von der anderen Seite akzeptiert oder die Conversation/Verbindung zwischen den Partnern kommt nicht zustande.

Die Verschlüsselung wird immer conversation- bzw. Verbindungs-spezifisch ausgehandelt. Eine Nachrichten-spezifische Verschlüsselung über die Programmschnittstelle ist nicht möglich.

In der Konfiguration der UTM-Anwendung können Sie jedem Client und jedem Service eine Verschlüsselungsebene zuordnen. Die Verschlüsselungsebene gibt an, ob Nachrichten vom Client verschlüsselt werden müssen oder nicht. Die Verschlüsselungsebenen werden mit der KDCDEF-Option ENCRYPTION-LEVEL in der TAC-, der PTERM- und der TPOOL-Anweisung festgelegt.

Die Verschlüsselungsebenen können bei openUTM dazu verwendet werden, sowohl den Zugang durch Clients als auch den Zugriff auf bestimmte Services zu kontrollieren.

5.9.3.1 Zugangsschutz

In der UTM-Konfiguration kann man für jeden Client (PTERM) und jede Client-Gruppe (LTERM-Pool; TPOOL) eine Verschlüsselungsebene festlegen. Die Verschlüsselungsebene gibt an, ob und wie Clients Nachrichten verschlüsseln müssen oder dürfen. Auf diese Weise kann sich eine UTM-Anwendung vor dem Zugang über unsichere Clients schützen.

Die Verschlüsselungsebene für einen Client legen Sie bei der KDCDEF-Generierung in der PTERM-Anweisung des Clients fest:

```
PTERM ...,ENCRYPTION-LEVEL=
```

Für Clients, die sich über einen LTERM-Pool an die Anwendung anschließen, legen Sie die Verschlüsselungsebene wie folgt fest:

```
TPOOL ...,ENCRYPTION-LEVEL=
```

Es gibt folgende Verschlüsselungsebenen:

1. openUTM fordert die Verschlüsselung vom Client an.
   Der Client muss auf jeden Fall verschlüsseln, sonst bekommt er keinen Zugang zur UTM-Anwendung. Dabei wird die Mindestlänge des verwendeten RSA-Schlüssels vorgegeben. Unterstützt der Partner keine Verschlüsselung oder kann er den RSA-Schlüssel mit der geforderten Schlüssellänge nicht verwenden, dann kann er keine Verbindung zur UTM-Anwendung aufbauen.

Diesen Fall generieren Sie mit folgenden Varianten:

`ENCRYPTION-LEVEL=1` (RSA-Schlüssellänge 200 bit, DES-Verfahren)
`ENCRYPTION-LEVEL=2` (RSA-Schlüssellänge 512 bit, AES-Verfahren)
`ENCRYPTION-LEVEL=3` (RSA-Schlüssellänge 1024 bit, AES-Verfahren)
`ENCRYPTION-LEVEL=4` (RSA-Schlüssellänge 2048 bit, AES-Verfahren)

2. openUTM fordert keine Verschlüsselung an, der Client kann festlegen, ob auf der Verbindung verschlüsselt wird oder nicht.
   Der Client wird zwar auch ohne Verschlüsselung zugelassen, er muss jedoch verschlüsseln, wenn es ein Service explizit verlangt (siehe nächster Abschnitt „Zugriffsschutz“).
   Diesen Fall generieren Sie mit:

   `ENCRYPTION-LEVEL=NONE`

3. Der Client wird als vertrauenswürdig eingestuft (*trusted Client*). Auf Verbindungen zu solchen Clients wird nicht verschlüsselt. Ein vertrauenswürdiger Client kann auch ohne Verschlüsselung „geschützte“ Services aufrufen (siehe nächster Abschnitt „Zugriffsschutz“).
   Clients sollten Sie nur dann als vertrauenswürdig generieren, wenn sichergestellt ist, dass die Kommunikation über sichere Leitungen erfolgt.
   Diesen Fall generieren Sie mit:

   `ENCRYPTION-LEVEL=TRUSTED`

X/W X/W

Unix- und Windows-Systeme
Lokale UPIC-Clients (Typ UPIC-L) sind immer trusted Clients.

5.9.3.2 Zugriffsschutz

Mit Hilfe der Verschlüsselungs-Funktionen können Sie einzelne Services vor dem Zugriff über unsichere Clients schützen. Ein Client darf auf solche „geschützten“ Services nur dann zugreifen, wenn er entweder als vertrauenswürdig eingestuft ist oder wenn er mit dem geforderten Verfahren verschlüsseln kann.

Einen Service können Sie schützen, indem Sie dem zugehörigen Vorgangs-TAC die Verschlüsselungsebene 1 oder 2 zuweisen:

`TAC ...,ENCRYPTION-LEVEL=1` (Verschlüsselung nach dem DES-Verfahren)

`TAC ...,ENCRYPTION-LEVEL=2` (Verschlüsselung nach dem AES-Verfahren)

Ist ein Service auf diese Weise geschützt, dann gilt:

- Von einem vertrauenswürdigen Client kann ein solcher Service gestartet werden, ohne dass eine Verschlüsselung zum Einsatz kommt.
- Für nicht vertrauenswürdige Clients (*non-trusted* Clients), die die Verschlüsselungsfunktion unterstützen, wird der zu dem Transaktionscode gehörende Vorgang nur gestartet, wenn der Client die Eingabe-Nachricht mit dem geforderten Verfahren verschlüsselt übertragen hat. Andernfalls gilt:
  - Bei UPIC-Clients wird der Aufbau der Conversation durch openUTM abgelehnt.
  - Bei VTSU-Partnern führt dies zu BADTAC oder es wird die Meldung K009 ausgegeben.

  Wird der Vorgang über einen Transaktionscode ohne Benutzerdaten aufgerufen (z.B. bei Terminal-Emulationen über eine Funktionstaste) oder durch Vorgangskettung gestartet, dann wird der Vorgang auch ohne Verschlüsselung gestartet. openUTM verschlüsselt dann alle Dialog-Ausgabe-Nachrichten an den Client. Bei Mehrschritt-Vorgängen erwartet openUTM alle weiteren Eingabe-Nachrichten vom Client verschlüsselt.
  Enthält die Eingabe-Nachricht unverschlüsselte Benutzerdaten, dann wird der Vorgang abnormal beendet.
- Versucht ein nicht vertrauenswürdiger Client, der die Verschlüsselungs-Funktionen nicht unterstützt, den Service zu starten, dann wird die Conversation zum UPIC-Client abgelehnt. Wird versucht, den Service über Vorgangskettung zu starten, so wird der Vorgang abnormal beendet.
  Wird der Service über einen VTSU-Partner (Terminal-Emulation) aufgerufen, dann führt dies zu BADTAC oder es wird die Meldung K009 ausgegeben.

Die Verschlüsselung ist optional, wenn Sie einen Vorgangs-TAC wie folgt generieren (Standard):

```
TAC ...,ENCRYPTION-LEVEL=NONE
```

Informationen zur Verschlüsselung an der KDCS-Programmschnittstelle

Sie haben auch die Möglichkeit eigene Teilprogramme zu schreiben, die eine Prüfung der Zugriffsberechtigung durchführen. Dazu werden an der Programmschnittstelle beim INIT PU-Aufruf Daten bezüglich Verschlüsselung angezeigt. Angezeigt werden:

- die Verschlüsselungsebenen, die für den Client und Transaktionscode generiert sind
- ob für die Conversation Verschlüsselung ausgehandelt wurde
- ob der Client prinzipiell Verschlüsselung unterstützt
- ob die letzte Eingabe-Nachricht verschlüsselt war

5.9.4 RSA-Schlüsselpaar erzeugen und öffentlichen Schlüssel auslesen

Aus Sicherheitsgründen sollten Sie das RSA-Schlüsselpaar Ihrer UTM-Anwendung in regelmäßigen Abständen durch ein neues RSA-Schlüsselpaar ersetzen. Die Programmschnittstelle zur Administration und das Administrationstool openUTM-WinAdmin bieten entsprechende Funktionen an.

Siehe openUTM-Handbuch „Anwendungen administrieren“; KDCADMI-Operationscode KC_ENCRYPT oder Hilfe zu openUTM-WinAdmin.

Mit Hilfe der Administration können Sie ein neues Schlüsselpaar erzeugen, den öffentlichen Schlüssel auslesen und das neue Schlüsselpaar aktivieren. Erst nach dem Aktivieren wird das neue Schlüsselpaar von der UTM-Anwendung für die Verschlüsselung eingesetzt. Ein aktiviertes Schlüsselpaar kann per Administration auch gelöscht werden.

Um die Sicherheit der Daten auf einer Verbindung weiter zu erhöhen, können Sie den öffentlichen Schlüssel des RSA-Schlüsselpaares auslesen, über ein eigenes Verfahren an den Client übertragen und dort hinterlegen. Erst dann sollten Sie das neue RSA-Schlüsselpaar aktivieren. Mit Hilfe des hinterlegten öffentlichen RSA-Schlüssels kann der Client verifizieren, ob der über die Verbindung zur UTM-Anwendung empfangene öffentliche Schlüssel auch wirklich von der UTM-Anwendung stammt.

5.10 Datenbankkopplung definieren

Bei der Konfiguration der Anwendung müssen Sie über KDCDEF-Steueranweisungen die Datenbanksysteme beschreiben, mit denen die Anwendung zusammenarbeiten soll.

Bei einer UTM-Anwendung, die mit einer Datenbank gekoppelt werden soll, sind beim Binden und Starten zusätzliche Parameter anzugeben. Sehen Sie dazu das openUTM-Handbuch „Einsatz von openUTM-Anwendungen“.
Die übrige UTM-Generierung wird von der Kopplung nicht beeinflusst.

5.10.1 Datenbankkopplung unter BS2000/OSD

B openUTM unterstützt die Koordination mit folgenden Datenbanksystemen:

B – UDS/SQL
B – SESAM/SQL
B – XA
B – CIS
B – PRISMA
B – LEASY (das Dateisystem LEASY verhält sich gegenüber openUTM wie ein Datenbanksystem)

B Eine UTM-Anwendung kann mit maximal 2 - auf Sonderfreigabe auch mit bis zu 8 - verschiedenen Datenbanken koordiniert zusammenarbeiten. Jedes Datenbanksystem wird dazu mit einer DATABASE-Anweisung im KDCDEF definiert.

B **DATABASE-Anweisung** auf Seite 317
Definition der Datenbank, mit der die UTM-Anwendung zusammenarbeitet:

B • ENTRY=

B Entry-Namen der unterstützten Datenbank, den Sie der Tabelle auf Seite 317 entnehmen können.

B • LIB=

B Objektmodulbibliothek, aus der das Verbindungsmodul zu dem Datenbanksystem nachgeladen werden soll.

B • TYPE=

B Typkennzeichen des Datenbanksystems.
B – Mit TYPE=DB können Datenbanksysteme angeschlossen werden, die nicht in obiger Liste enthalten sind, aber die Schnittstelle IUTMDB unterstützen.
B – Mit TYPE=XA wird die Kopplung zu einer XA-Resource generiert.

5.10.2 Kopplung mit einem Resource Manager unter Unix- und Windows-Systemen

X/W Die Kopplung mit Resource Managern (z.B. Datenbanksysteme) erfolgt über die von X/Open genormte XA-Schnittstelle. Sie koordiniert die Transaktionen von openUTM mit den Diensten des Resource Managers. Die XA-Schnittstelle wird in der CAE-Version der XA-Schnittstelle (XA-CAE) unterstützt.

X/W openUTM kann mit allen Resource Managern gekoppelt werden, die die XA-CAE-Schnittstelle anbieten, wie beispielsweise Oracle und INFORMIX.

X openUTM auf Unix-Systemen unterstützt darüber hinaus die Koordination mit allen anderen Datenbank-, Datei- und Queuing-Systemen, die die XA-Schnittstelle (X/Open-Standard) unterstützen: z.B. SYBASE, ADABAS, DB2, O_2, ISAM/XA, SQL-Server, IBM MQ Series.

W openUTM auf Windows-Systemen unterstützt Oracle.

X/W **RMXA-Anweisung** auf Seite 469
Den Resource Manager, mit dem openUTM gekoppelt werden soll, und die Version der XA-Schnittstelle, über die die Kopplung erfolgen soll, müssen Sie bei der Generierung mit der RMXA-Anweisung definieren:

X/W
- XASWITCH=

  X/W Name der xa_switch_t-Struktur des Resource Managers, die openUTM bekannt gemacht wird.

X/W
- SPEC=C

  X/W Es wird die CAE-Spezifikation der XA-Schnittstelle für die Kopplung mit dem Resource Manager benutzt.

X/W Für den gekoppelten Betrieb von openUTM mit XA ist Folgendes zu beachten:

X/W
- Innerhalb einer UTM-Anwendung können auch mehrere Resource Manager (d.h. Datenbanksysteme) bedient werden.

X/W
- Es ist nicht erlaubt, mehrere Resource Manager mit identischen Generierungsparametern zu generieren. D.h. innerhalb eines Generierungslaufs dürfen nicht mehrere RMXA-Anweisungen mit demselben Namen für die xa_switch_t-Struktur des Resource Managers angegeben werden.

X/W
X/W
X/W
X/W
X/W
- Der gleichzeitige Betrieb mehrerer Instanzen (Datenbanken) eines Resource Managers (Datenbanksystem) ist möglich, sofern der Resource Manager den Multi-Instanzen-Betrieb unterstützt. Mit welchen Datenbanken die UTM-Anwendung gekoppelt wird, bestimmen Sie mit entsprechenden Startparametern der Anwendung. Für den Multi-Instanzen-Betrieb müssen Sie beim Start mehrere Openstrings angeben.

X/W
X/W
X/W
X/W
X/W
Im Folgenden ist für Oracle, INFORMIX und Microsoft SQL Server beschrieben, wie Sie die Kopplung Ihrer UTM-Anwendung mit dem Resource Manager generieren müssen. Die hier angegebenen Datenbank-spezifischen Namen (xa_switch_t-Struktur) können sich ändern, deshalb sollten Sie die Richtigkeit der Angaben in der Dokumentation zu den jeweiligen Datenbanksystemen überprüfen.

X/W *Kopplung mit Oracle*

X/W
```
RMXA XASWITCH=xaosw,SPEC=C
```
X/W bzw.
X/W
```
RMXA XASWITCH=xaoswd,SPEC=C
```

X *Kopplung mit INFORMIX*

X
```
RMXA XASWITCH=infx_xa_switch,SPEC=C
```

5.11 Internationalisierung der Anwendung - XHCS Unterstützung (BS2000/OSD)

B Eine UTM-Anwendung in BS2000/OSD kann so programmiert werden, dass verschiedensprachige Kommunikationspartner die Nachrichten von den Teilprogrammen in ihrer jeweiligen Landessprache erhalten. Selbst regionale Besonderheiten innerhalb einer Sprache können dabei berücksichtigt werden. Datumsangaben, Uhrzeit, Maßangaben oder Währungssymbole können den landessprachlichen Konventionen entsprechend dargestellt werden.

B Zur Darstellung der Schrift- und Sonderzeichen der einzelnen Sprachen an einem Terminal oder Drucker werden u.U. verschiedene erweiterte Zeichensätze (8-Bit-Codes) benötigt. Mit Hilfe des BS2000-Softwareprodukts XHCS (E**x**tended **H**ost **C**ode **S**upport) können in einem BS2000/OSD-System gleichzeitig mehrere erweiterte Zeichensätze verwendet werden. openUTM unterstützt die Funktionen von XHCS. Damit können Sie den UTM-Objekten jeweils eine bestimmte Sprachumgebung - auch Locale genannt - zuordnen, d.h. Sie können der Anwendung ein Standard-Locale zuordnen. Einzelnen Benutzern und LTERM-Partnern, über die sich Clients an die Anwendung anschließen, werden jeweils spezifische Locale zugeordnet, die bei der Aufbereitung der Nachrichten verwendet werden.

B Um die Mehrsprachigkeit in UTM-Anwendungen zu realisieren, bietet openUTM folgende Funktionen an:

- B Bei der Generierung der Anwendung können Sie der Anwendung, den Benutzerkennungen, den LTERM-Partnern und den LTERM-Partnern der LTERM-Pools eigene Sprachen und die für die Ausgabe zu verwendenden Zeichensätze zuordnen. Dazu definieren Sie für die Objekte so genannte Locale, die Sprachumgebung und Zeichensatz festlegen.
- B Es können Locale für Benutzermeldungsmodule definiert werden, die landessprachliche Besonderheiten berücksichtigen. Diese landessprachlichen Meldungsmodule werden Benutzern und LTERM-Partnern zugeordnet, die in Sprach- und Territorialkennzeichen mit dem Locale des landesprachlichen Meldungsmoduls übereinstimmen. Sehen Sie dazu auch Abschnitt „UTM-Meldungen“ auf Seite 183.
- B Während des Anwendungsbetriebs kann ein Benutzer die Zuordnung von Sprache und Zeichensatz für seine Benutzerkennung ändern. Dazu bietet die KDCS-Schnittstelle den Aufruf SIGN CL.
- B Mit Hilfe der Varianten INIT PU und INFO LO der Funktionsaufrufe INIT und INFO kann ein UTM-Teilprogramm Sprache und Zeichensatz der Benutzerkennung, der Anwendung, eines bestimmten LTERM-Partners oder von LTERM-Partnern in einem Pool lesen. Das Teilprogramm erhält so die Informationen über die vom Terminal unterstützten Zeichensätze und den Zeichensatz der Eingabe-Nachricht. Mit diesen Informationen kann es die Eingaben des Benutzers richtig interpretieren und dem Benutzer Nachrichten in der richtigen Sprache und mit dem passenden Zeichensatz senden.

B
B
B
B
B
B
- Ist die Nachricht eines Teilprogramms an ein Terminal oder einen Drucker gerichtet, dann übergibt openUTM die logische Nachricht des Teilprogramms zusammen mit dem Namen des Zeichensatzes, der bei der Aufbereitung verwendet werden soll, an VTSU-B. VTSU-B bereitet die Nachricht zur Ausgabe an Terminal oder Drucker auf. Welcher Zeichensatz für die Nachrichtenaufbereitung das ist, ist im openUTM-Handbuch „Anwendungen programmieren mit KDCS“ beschrieben.

B
B
Ist der Auftraggeber in einem Vorgang ein Partnerprogramm, dann wird die logische Nachricht ohne Aufbereitung an den Auftraggeber übergeben.

B
B
B
B
B
B
- Das Teilprogramm kann mit INFO LO von openUTM Information über Sprache und Zeichensatz des LTERM-Partners und der Zeichensätze anfordern, die vom Terminal/Drucker unterstützt werden, das diesem LTERM-Partner zugeordnet ist. Der Zeichensatz, der bei der Aufbereitung der Nachricht zur Ausgabe an Terminal/Drucker verwendet wird, muss zu einem der von diesem Terminal/Drucker unterstützten Zeichensätze kompatibel sein.

B
Bevor auf diese Funktionen eingegangen wird, werden spezielle XHCS-Begriffe erläutert.

5.11.1 XHCS-Begriffsdefinitionen

B
ISO-Zeichensätze, Variantennummern

B
B
B
B
B
B
B
B
In ISO 8859 sind verschiedene erweiterte Zeichensätze für verschiedene Sprachräume normiert, beispielsweise ISO 8859-1, ISO 8859-2 usw. Die Nummern am Ende (-1, -2 usw.) heißen *Variantennummern*. Ein erweiterter Zeichensatz enthält alle Zeichen, die zur Darstellung der Sprachen eines Sprachraums benötigt werden.
ISO 8859-Codes sind Erweiterungen des ASCII-Codes ISO 646. Sie werden z.B. von Terminals und Unix-Systemen verwendet. Alle ISO 8859-Zeichensätze enthalten in der niederwertigen Hälfte der Codetabelle den ASCII-Code als gemeinsamen Teil.

B
EBCDIC-Zeichensätze

B
B
B
B
B
B
B
B
Im BS2000-Betriebssystem wird mit EBCDIC-Zeichensätzen gearbeitet. Zu jedem ISO 8859-Code existiert eine Erweiterung des EBCDIC.DF.03-IRV bzw. -DRV. Der EBCDIC.DF.03-IRV ist die Internationale Referenz-Version und EBCDIC.DF.03-DRV die Deutsche Referenz-Version des nicht erweiterten EBCDIC-Codes. Beide Codes enthalten den EBCDIC-Kern als gemeinsamen Zeichenvorrat und unterscheiden sich nur in einigen Symbolen. Die Erweiterungen dieser EBCDIC-Zeichensätze heißen EBCDIC.DF.04-1, EBCDIC.DF.04-2 bis EBCDIC.DF.04-10.

Kompatible Zeichensätze

Erweiterte ISO- und EBCDIC-Zeichensätze mit derselben Variantennummer sind *kompatibel*, d.h. sie enthalten dieselben Zeichen. Die einzelnen Zeichen stehen jedoch an anderen Codeplätzen innerhalb der Codetabelle. Die Codes sind mit Hilfe von Umsetztabellen ineinander überführbar.

Der BS2000-Systemverwalter kann mit Hilfe von XHCS die EBCDIC-Zeichensätze modifizieren, indem er den einzelnen Zeichen eines Zeichensatzes andere Codeplätze in der Codetabelle zuweist. Die Gesamtmenge der Zeichen bleibt dabei erhalten. Die modifizierten EBCDIC-Zeichensätze heißen *kompatibel* zu dem EBCDIC.DF.04-n-Zeichensatz, aus dem sie erzeugt wurden.

Referenzcode

XHCS fasst jeweils alle zueinander kompatiblen Zeichensätze des Systems zu einer Gruppe zusammen. Zu einer Gruppe gehören also jeweils eine ISO-Variante und die zu ihr kompatiblen EBCDIC-Zeichensätze. Der EBCDIC.DF.04-n-Zeichensatz der Gruppe ist der Referenzcode der Gruppe. Alle Zeichensätze einer Gruppe können mit Hilfe von XHCS in den Referenzcode der Gruppe umgesetzt werden.

Coded-Character-Set-Name (CCS-Name)

Jedem Zeichensatz, der im System verwendet wird, wird ein maximal acht Zeichen langer Name, der CCS-Name oder CCSN, zugeordnet. Über den CCS-Namen wird der Zeichensatz im System eindeutig identifiziert. Die CCS-Namen der Referenzcodes werden von XHCS vorgegeben. EBCDIC.DF.04-1 hat z.B. den CCS-Namen EDF041.

Eine Liste der CCS-Namen für die in Ihrem BS2000/OSD verfügbaren Zeichensätze erhalten Sie mit Hilfe des EDT ab Version V16.4. Rufen Sie dazu den EDT auf und geben Sie die EDT-Anweisung `@SHOW CCS` ein. Der EDT liefert dann eine Liste der verfügbaren Zeichensätze.

System-Standard-Code

Der BS2000-Systemverwalter kann mehrere erweiterte Zeichensätze (auch für verschiedene ISO-Varianten) definieren, die von den Systemkomponenten gleichzeitig verwendet werden können.

Einen dieser Zeichensätze kann der Systemverwalter als System-Standard-Code definieren. Den aktuell eingestellten System-Standard-Code können Sie der Ausgabe des Kommandos /SHOW-SYSTEM-PARAMETERS PAR=*ALL entnehmen. Er wird im Parameter HOSTCODE angegeben.

Standard-Anwenderzeichensatz

Der BS2000-Systemverwalter kann jeder BS2000-Benutzerkennung einen der im System definierten Zeichensätze als Standard-Anwenderzeichensatz zuordnen. Ist für die BS2000-Benutzerkennung ein Standard-Anwenderzeichensatz definiert, so wird sein CCS-Name im Ausgabefeld CODED-CHARACTER-SET des Kommandos /SHOW-USER-ATTRIBUTES ausgegeben.

Weitere Information zu XHCS finden Sie im Benutzerhandbuch „XHCS 8-bit-Code-Verarbeitung im BS2000/OSD - Internationalisierung“.

5.11.2 Sprachumgebung festlegen - Locale definieren

Bei der Generierung einer UTM-Anwendung kann für die UTM-Anwendung, für jeden LTERM-Partner, für alle LTERM-Partner in einem LTERM-Pool und für jede Benutzerkennung eine eigene Sprachumgebung festgelegt werden. Dazu ordnen Sie der Anwendung und den einzelnen Objekten ein Tripel bestehend aus Sprachkennzeichen, Territorialkennzeichen und Name eines Zeichensatzes zu, das als Locale bezeichnet wird. Das Locale wird wie folgt angegeben:

```
LOCALE=( lang_id,terr_id,ccsname )
```

lang_id Das Sprachkennzeichen *lang_id* kennzeichnet die Sprache, in der der Benutzer von den UTM-Teilprogrammen bedient werden soll. Das Sprachkennzeichen darf max. 2 Byte lang sein. Die Bezeichnung einer Sprache ist frei wählbar.

terr_id Das Territorialkennzeichen *terr_id* dient dazu, regional bedingt Unterschiede innerhalb einer Sprache (Englisch in England und Amerika) oder unterschiedliche Währungs- und Maßeinheiten in den Ländern (Dollar, englisches Pfund) berücksichtigen zu können. Das Territorialkennzeichen darf max. 2 Byte lang sein und ist frei wählbar.

ccsname Der Zeichensatzname *ccsname* gibt an, welcher Zeichensatz bei der Aufbereitung einer Nachricht für die Ausgabe am Terminal verwendet werden kann. Als Zeichensatzname ist der CCS-Name eines Zeichensatzes anzugeben, der im BS2000-System definiert ist. CCS-Namen werden vom BS2000-Systemverwalter vergeben.

Wenn alle Benutzer aus einem Sprachraum kommen, z.B. Westeuropa, dann genügt es, der UTM-Anwendung einen erweiterten Zeichensatz zuzuordnen. Die Verwendung von Benutzer-spezifischen Zeichensätzen ist nur nötig, wenn die verschiedenen Benutzer einer Anwendung Sprachen sprechen, die nicht alle mittels eines erweiterten Zeichensatzes dargestellt werden können.

Voraussetzung für die Unterstützung von erweiterten Zeichensätzen ist, dass auf dem Rechner, auf dem die UTM-Anwendung läuft, das Subsystem XHCS verfügbar ist. Für alle in der UTM-Anwendung generierten Zeichensatznamen müssen in XHCS zugehörige EBCDIC-Zeichensätze definiert sein. Zudem müssen die Terminals einen zum jeweiligen EBCDIC-Zeichensatz kompatiblen ISO-Zeichensatz unterstützen. Nur bestimmte Terminaltypen und Drucker unterstützen 8-Bit-Zeichensätze.

Anwendungs-spezifische Sprachumgebung - Standard-Sprachumgebung

MAX-Anweisung auf Seite 368
Der UTM-Anwendung ordnen Sie das Locale bei der Generierung mit Hilfe der MAX-Anweisung zu:

- LOCALE=

  Das für die Anwendung generierte Locale wird jeder Benutzerkennung, jedem LTERM-Partner und jedem LTERM-Pool als Standardwert für die Sprachumgebung zugeordnet. Diese Standardeinstellung ist wirksam, solange für diese Objekte kein eigenes Locale definiert wird.

Benutzer-spezifische Sprachumgebung

USER-Anweisung auf Seite 538
Einer Benutzerkennung ordnen Sie ein Locale mit Hilfe der USER-Anweisung zu:

- LOCALE=

  Der einer Benutzerkennung zugeordnete Zeichensatz wird bei der Ausgabe von Dialog-Nachrichten an den Bildschirm verwendet (siehe dazu „Zeichensätze für die Nachrichtenaufbereitung“ im openUTM-Handbuch „Anwendungen programmieren mit KDCS“).

B **LTERM-Partner-spezifische Sprachumgebung**

B

LTERM-Anweisung auf Seite 354 und **TPOOL-Anweisung** auf Seite 518
Einem LTERM-Partner, über den sich ein Terminal oder Drucker an die Anwendung anschließt, ordnen Sie mit der LTERM-Anweisung ein Locale zu. Für einen LTERM-Pool wird für alle LTERM-Partner dieses Pools mit der TPOOL-Anweisung ein Locale festgelegt:

- LOCALE=

  Der für den LTERM-Partner definierte Zeichensatz wird für die Ausgabe von Asynchron-Nachrichten verwendet (siehe dazu „Zeichensätze für die Nachrichtenaufbereitung“ im openUTM-Handbuch „Anwendungen programmieren mit KDCS“).

  Das LTERM-Partner-spezifische Locale wird z.B. auch im ersten Teil des Anmelde-Vorgangs verwendet, wenn der Benutzer noch nicht angemeldet, d.h. die Benutzer-spezifische Sprachumgebung noch nicht eingerichtet ist.

B **Beispiel**

In der Anwendung wird das Sprachkennzeichen `DE` für deutsch verwendet. Um in Nachrichten an Benutzer in Deutschland und der Schweiz die unterschiedlichen Währungseinheiten (Euro und Franken) berücksichtigen zu können, werden die Territorialkennzeichen `DE` für Deutschland und `CH` für Schweiz definiert. Zur Ausgabe von Nachrichten kann der EBCDIC-Zeichensatz EBCDIC.DF.04-1 verwendet werden. Sein CCS-Name ist `EDF041`.

- Für die Anwendung kann das Locale für Benutzer in Deutschland als Standard-Sprachumgebung definiert werden. Dazu geben Sie in der MAX-Anweisung Folgendes an:

  ```
  MAX ..., LOCALE=(DE,DE,EDF041)
  ```

  Für Benutzer und Terminals in Deutschland, die sich über LTERM-Partner an die Anwendung anschließen, muss in diesem Fall kein eigenes Locale definiert werden.

- Sollen für Benutzer und Terminals in der Schweiz die Landes-spezifischen Besonderheiten berücksichtigt werden, muss Folgendes generiert werden:

  ```
  USER username ,..., LOCALE=(DE,CH,EDF041)
  LTERM ltermname ,..., LOCALE=(DE,CH,EDF041)
  ```

- Sie können aber auch über die DEFAULT-Anweisung für alle USER- und LTERM-Anweisungen das Locale (DE,DE,EDF041) einstellen:

```
DEFAULT USER LOCALE=(DE,DE,EDF041)
DEFAULT LTERM LOCALE=(DE,DE,EDF041)
```

Für Benutzer und Terminals in der Schweiz, die sich über LTERM-Partner an die Anwendung anschließen, generieren Sie dann zusätzlich Folgendes:

```
USER username ,..., LOCALE=(,CH)
LTERM ltermname ,..., LOCALE=(,CH)
```

5.11.3 Zeichensatznamen für Editprofile und Formate

Zusätzlich zur benutzer- und LTERM-Partner-spezifischen Zuordnung von Zeichensatznamen kann jedem in der Anwendung definierten Editprofil ein eigener Zeichensatzname zugeordnet werden.

Bei der Erstellung von Formaten mit FHS/IFG kann jedem Format der Name eines Zeichensatzes zugeordnet werden. Sehen Sie dazu auch die Handbücher zu FHS und IFG.

Welcher der generierten Zeichensatznamen (anwendungs-, benutzer-, LTERM-Partner-spezifischer Zeichensatz oder der einem Editprofil bzw. einem Format zugeordnete Zeichensatzname) bei der Aufbereitung einer Nachricht zur Ausgabe am Bildschirm oder Drucker verwendet wird, ist im openUTM-Handbuch „Anwendungen programmieren mit KDCS“ beschrieben.

5.12 Zugangskontrolle mit Kerberos generieren (BS2000/OSD)

B Für die Generierung der Zugangskontrolle mit dem verteilten Authentifizierungsdienst Kerberos sind folgende Generierungsanweisungen relevant:

B
- LTERM KERBEROS-DIALOG=

  B Wenn Sie LTERM KERBEROS-DIALOG=YES angeben, wird beim Verbindungsaufbau für Terminals, die Kerberos unterstützen und die sich direkt (nicht über OMNIS) über diesen LTERM-Partner an die Anwendung anschließen, ein Kerberos-Dialog durchgeführt (siehe Seite 359ff).

B
- TPOOL KERBEROS-DIALOG=

  B Wenn Sie TPOOL KERBEROS-DIALOG=YES angeben, wird beim Verbindungsaufbau für Terminals, die Kerberos unterstützen und die sich direkt (nicht über OMNIS) über diesen Terminalpool an die Anwendung anschließen, ein Kerberos-Dialog durchgeführt (siehe Seite 525ff).

B openUTM speichert die Kerberos-Information in der Länge ab, die sich aus dem Maximum der bei MAX PRINCIPAL-LTH und MAX CARDLTH generierten Längen ergibt (siehe Seite 376 und Seite 392). Wenn die Kerberos-Information länger ist, wird sie auf diese Länge verkürzt abgespeichert.

6 Generierungstool KDCDEF

Wenn Sie eine UTM-Anwendung generieren oder eine bereits existierende UTM-Anwendung anpassen wollen, definieren Sie zunächst die Konfiguration der UTM-Anwendung mit KDCDEF-Steueranweisungen und generieren anschließend mit dem Generierungstool KDCDEF die KDCFILE und die Anwendungs-Komponente KDCROOT, aus der Sie das UTM-Anwendungsprogramm erzeugen. Siehe dazu auch Kapitel „Einführung in die Generierung“ auf Seite 29.

Die Konfiguration einer Anwendung können Sie auch dynamisch im laufenden Betrieb ändern. Dazu reservieren Sie bereits bei der Generierung mit der RESERVE-Anweisung Tabellenplätze für UTM-Objekte. Sie können auf diese Weise Clients, Drucker, Benutzerkennungen und Services, KSETs, LTACs, CONs und LSESes „on-the-fly“ in die Konfiguration neu aufnehmen oder löschen, die Verfügbarkeit wird dadurch nicht beeinträchtigt. Das dynamische Eintragen von Objekten wird ausführlich im openUTM-Handbuch „Anwendungen administrieren“ beschrieben.

Die Konfigurationsinformationen in der KDCFILE einer dynamisch konfigurierten Anwendung können Sie mit der Anweisung CREATE-CONTROL-STATEMENTS während des KDCDEF-Laufs aus der KDCFILE auslesen und in Steueranweisungen umwandeln lassen. Diese Funktion heißt inverser KDCDEF. Die so erzeugten Steueranweisungen werden in eine Datei geschrieben, die Sie unmittelbar als Eingabedatei für den KDCDEF-Lauf wiederverwenden können. Siehe dazu den Abschnitt „Inversen KDCDEF starten“ auf Seite 267.

6.1 ROOT-Tabellen-Source und KDCFILE erstellen

Das Generierungstool KDCDEF erstellt anhand der Konfigurationsinformationen in den KDCDEF-Steueranweisungen die KDCFILE, die alle Konfigurations- und Verwaltungsdaten enthält, und die ROOT-Tabellen-Source für die Main Routine KDCROOT.

Die KDCFILE und die ROOT-Tabellen-Source können Sie zusammen in einem KDCDEF-Lauf oder einzeln in getrennten KDCDEF-Läufen generieren. Sie legen dies mit der KDCDEF-Anweisung OPTION...,GEN= fest.

Alle KDCDEF-Anweisungen, mit denen Sie die Konfiguration einer UTM-Anwendung beschreiben können, finden Sie in den folgenden Tabellen, jeweils zu Funktionsgruppen zusammengefasst.

6.1.1 Anweisungen zur Steuerung des KDCDEF-Laufs

	Anweisung	Funktion
	EJECT	Seitenvorschub im Protokoll veranlassen
	END	Eingabe für KDCDEF beenden
	OPTION	KDCDEF-Lauf steuern
	REMARK oder *	Kommentarzeile schreiben
B	***zusätzliche Anweisung unter BS2000/OSD***	
B	DEFAULT	Standardwerte definieren

6.1.2 Anweisungen zum Erzeugen der ROOT-Tabellen-Source

	Anweisung	Funktion
	AREA	Namen zusätzlicher Datenbereiche festlegen
	EXIT	Event Exits definieren
	FORMSYS	Formatierungssystem beschreiben
	MAX	UTM-Anwendungsparameter definieren
	MESSAGE	Meldungsmodul beschreiben
	PROGRAM	Teilprogramme definieren
	RESERVE	Tabellenplätze für dynamisch eintragbare Objekte reservieren
	ROOT	Namen für ROOT-Tabellen-Source vergeben
B	***zusätzliche Anweisungen unter BS2000/OSD***	
B	DATABASE	Datenbanksystem beschreiben (BS2000/OSD)
B	LOAD-MODULE	Lademodule für BLS beschreiben
B	MPOOL	Common Memory Pool beschreiben
B	TCBENTRY	Gruppe von TCB-Entries definieren
X/W	***zusätzliche Anweisungen unter Unix-, Windows-Systemen***	
X/W X/W	RMXA	Auf Unix- und Windows-Systemen Namen für Resource Manager angeben (Datenbankkopplung über die X/Open XA-Schnittstelle)
X/W	SHARED-OBJECT	Shared Objects/DLLs für den Programmaustausch definieren

6.1.3 Basisanweisungen zum Erzeugen einer KDCFILE

	Anweisung	Funktion
	ACCOUNT	Abrechnungsparameter für UTM-Accounting festlegen
	BCAMAPPL	Weitere Anwendungsnamen für parallele Verbindungen definieren
	CLUSTER	Globale Eigenschaften einer UTM-Cluster-Anwendung definieren
	CLUSTER-NODE	Knoten-Anwendung eines UTM-Clusters definieren
	CREATE-CONTROL-STATEMENTS	Aus der vorhandenen KDCFILE Steueranweisungen für erneuten KDCDEF-Lauf erzeugen
	KSET	Keyset definieren
	LTERM	LTERM-Partner als logischen Anschlusspunkt für Clients und Drucker definieren
	MAX	UTM-Anwendungsparameter definieren
	MESSAGE	Meldungsmodul beschreiben
	MSG-DEST	Benutzer-Meldungsziele definieren
	PROGRAM	Namen und Eigenschaften der Teilprogramme festlegen
	PTERM	Clients und Drucker definieren
	QUEUE	Tabelleneinträge für Temporäre Queues reservieren
	RESERVE	Tabellenplätze für dynamisch eintragbare Objekte reservieren
	SFUNC	Sonderfunktionen der F- und K-Tasten festlegen
	SIGNON	Anmeldeverfahren steuern
	TAC	Namen und Eigenschaften von Transaktionscodes festlegen
	TACCLASS	Prozesszahl für TAC-Klasse festlegen
	TAC-PRIORITIES	Prioritäten für die TAC-Klassen festlegen
	TLS	Namen von TLS-Blöcken festlegen
	TPOOL	LTERM-Pools definieren
	ULS	Namen von ULS-Blöcken festlegen
	USER	Benutzerkennungen definieren
B	***zusätzliche Basisanweisungen unter BS2000/OSD***	
B	DATABASE	Datenbanksystem beschreiben
B	EDIT	Edit-Optionen definieren
B	LOAD-MODULE	Lademodule beschreiben
B	MPOOL	Common Memory Pool beschreiben
B	MUX	Multiplexanschluss definieren
B	SATSEL	SAT-Protokollierung festlegen

	Anweisung	Funktion
X/W	*zusätzliche Basisanweisung unter Unix-, Windows-Systemen*	
X/W	SHARED-OBJECT	Shared Objects/DLLs für den Programmaustausch definieren

6.1.3.1 KDCFILE erzeugen - zusätzliche Anweisungen für verteilte Verarbeitung über LU6.1

Anweisung[1]	Funktion
BCAMAPPL	Weitere Anwendungsnamen für parallele Verbindungen definieren
CON	Logische Verbindung zu UTM-Partner-Anwendungen definieren
LPAP	LPAP-Partner als logischen Anschlusspunkt für UTM-Partner-Anwendung definieren
LSES	Sessionname für die Verbindung zweier UTM-Anwendungen festlegen
LTAC	Lokale Namen für TACs in UTM-Partner-Anwendungen vergeben
MASTER-LU61-LPAP	Master-LPAP eines LU6.1-LPAP-Bündels definieren
SESCHA	Sessioneigenschaften definieren
UTMD	Anwendungsglobale Werte festlegen

[1] Die Basisanweisung RESERVE kann auch für CON, LSES und LTAC verwendet werden

6.1.3.2 KDCFILE erzeugen - zusätzliche Anweisungen für verteilte Verarbeitung über OSI TP

	Anweisung[1]	Funktion
	ABSTRACT-SYNTAX	Abstrakte Syntax definieren
	ACCESS-POINT	OSI TP-Zugriffspunkt für lokale UTM-Anwendung einrichten
	APPLICATION-CONTEXT	Application Context definieren
	LTAC	Lokale Namen für TACs in UTM-Partner-Anwendungen vergeben
	MASTER-OSI-LPAP	Master-LPAP eines OSI-LPAP-Bündels definieren
	OSI-CON	Logische Verbindung zur Partner-Anwendung definieren
	OSI-LPAP	OSI-LPAP-Partner als logischen Anschlusspunkt der Partner-Anwendung definieren
	TRANSFER-SYNTAX	Transfersyntax definieren
	UTMD	Anwendungsglobale Werte und Adresse der lokalen UTM-Anwendung festlegen
X/W	*zusätzliche Anweisungen unter Unix-, Windows-Systemen*	
X/W	MAX XAPTPSHMKEY	Schlüssel für das XAPTP Shared Memory Segment definieren
X/W	MAX OSISHMKEY	Schlüssel für das Shared Memory Segment von OSS definieren
X/W	MAX OSI-SCRATCH-AREA	Arbeitsbereichgröße für die dynamische Ablage von Daten festlegen

[1] Die Basisanweisung RESERVE kann auch für LTAC verwendet werden

6.1.3.3 KDCFILE erzeugen - Anweisungen zum Erzeugen einer Cluster-Konfigurationsdatei

Anweisung	Funktion
CLUSTER	Globale Eigenschaften einer UTM-Cluster-Anwendung definieren
CLUSTER-NODE	Knoten-Anwendung einer UTM-Cluster-Anwendung definieren
MAX	UTM-Anwendungsparameter definieren

6.1.4 Auswirkungen der KDCDEF-Anweisungen auf die Generierungsobjekte

Nicht alle Anweisungen des Generierungstools KDCDEF wirken sich gleichermaßen auf die Generierungsobjekte KDCFILE und ROOT-Tabellen-Source aus. Welche Steueranweisung sich während des KDCDEF-Laufs auf welche Generierungsobjekte auswirkt, können Sie der folgenden Tabelle entnehmen:

KDCDEF-Steueranweisung	KDCFILE	ROOT-Tabellen	KDCDEF-Steuerung	Verteilte Verarbeitung über	
				LU6.1	OSI TP
ABSTRACT-SYNTAX	X				X
ACCESS-POINT	X				X
ACCOUNT	X				
APPLICATION-CONTEXT	X				X
AREA	X	X			
CLUSTER	X				
CLUSTER-NODE	X				
BCAMAPPL	X			X	
CON	X			X	
CREATE-CONTROL-STATEMENTS[1]					
EJECT			X		
END			X		
EXIT	X	X			
FORMSYS[2]	X	X			
KSET	X				
LPAP	X			X	
LSES	X			X	
LTAC	X			X	X
LTERM	X				
MASTER-LU61-LPAP	X			X	
MASTER-OSI-LPAP	X				X
MAX[3]	X	X			X
MESSAGE	X	X			
MSG-DEST	X				
OPTION[4]	X	X	X	(X)	(X)
OSI-CON	X				X

	KDCDEF-Steueranweisung	KDCFILE	ROOT-Tabellen	KDCDEF-Steuerung	Verteilte Verarbeitung über	
					LU6.1	OSI TP
	OSI-LPAP	X				X
	PROGRAM	X	X[5]			
	PTERM	X				
	QUEUE	X				
	REMARK			X		
	RESERVE	X	X[6]			
	ROOT		X			
	SESCHA	X			X	
	SFUNC	X				
	SIGNON	X				
	TAC	X				
	TACCLASS	X				
	TAC-PRIORITIES	X				
	TLS	X				
	TPOOL	X				
	TRANSFER-SYNTAX	X				X
	ULS	X				
	USER	X				
	UTMD	X			X	X
B	***BS2000/OSD-spezifische Anweisungen***					
B	DATABASE	X	X			
B	DEFAULT[7]	X	X	X		
B	EDIT	X				
B	LOAD-MODULE	X	X[8]			
B	MPOOL	X	X[9]			
B	MUX	X				
B	SATSEL	X				
B	TCBENTRY		X			
X/W	***Unix-, Windows-System-spezifische Anweisungen***					
X/W	RMXA		X			

	KDCDEF-Steueranweisung	KDCFILE	ROOT-Tabellen	KDCDEF-Steuerung	Verteilte Verarbeitung über	
					LU6.1	OSI TP
X/W	SHARED-OBJECT	X	X			

[1] Die CREATE-CONTROL-STATEMENTS-Anweisung erzeugt aus den Konfigurationsinformationen einer vorhandenen KDCFILE eine Eingabedatei mit KDCDEF-Steueranweisungen für einen erneuten KDCDEF-Lauf.

[2] Unter Windows-Systemen ist die Anweisung wirkungslos, es wird kein Formatierungssystem unterstützt.

[3] Die Operanden CLRCH=, KB=, NB=, und SPAB= wirken sich nur auf die Generierung der ROOT-Tabellen-Source aus. Die anderen Operanden wirken sich nur auf die Generierung der KDCFILE aus.

[4] Die Wirkung der OPTION-Anweisung auf die KDCFILE und die ROOT-Tabellen-Source ist abhängig von den Angaben bei OPTION...,GEN=.

[5] Nur bei Generierung einer UTM-Anwendung **ohne** Lademodule (in BS2000/OSD), Shared Objects (Unix-Systeme) bzw. DLLs (Windows-Systeme).

[6] Nur bei Generierung ohne Operand PROGRAM= und ohne Lademodule, Shared Objects bzw. DLLs.

[7] Die Wirkung der DEFAULT-Anweisung auf die KDCFILE und die ROOT-Tabellen-Source ist abhängig von der spezifizierten Unteranweisung.

[8] Nur bei Erweiterung der Generierung um *n* Lademodule.

[9] Nur bei Generierung ohne Lademodule.

Die Anweisungen MAX, CLUSTER und CLUSTER-NODE wirken auch auf die Cluster-Konfigurationsdatei. Wenn Sie in einer Neu-Generierung Parameter der CLUSTER- und/oder CLUSTER-NODE-Anweisung ändern, müssen Sie eine neue Cluster-Konfigurationsdatei erzeugen.

Mit der KDCDEF-Steueranweisung OPTION...,GEN= legen Sie fest, welche Objekte (KDCFILE, ROOT-Tabellen-Sourcen, Cluster-Konfigurationsdatei) das Generierungstool KDCDEF erstellen soll.

Nachdem Sie eine neue ROOT-Tabellen-Source erzeugt haben, müssen Sie diese übersetzen (In BS2000/OSD assemblieren). Ein Neu-Binden des Anwendungsprogramms ist nur erforderlich, falls der Tabellenmodul nicht dynamisch nachgeladen wird.
Wenn Sie nur die KDCFILE ändern, sind diese Schritte nicht nötig. Sie können die Anwendung beispielsweise mit einer neuen KDCFILE und der alten Main Routine KDCROOT betreiben.

6.2 Aufruf von KDCDEF und Eingabe der Steueranweisungen

6.2.1 KDCDEF starten und KDCDEF-Lauf durchführen

6.2.1.1 BS2000/OSD

Das Generierungstool KDCDEF starten Sie mit dem Kommando:

```
START-EXECUTABLE-PROGRAM FROM-FILE=*MODULE
          (LIBRARY=$userid.SYSLNK.UTM.060.UTIL,ELEMENT-OR-SYMBOL=KDCDEF)
```

KDCDEF liest die Generierungsanweisungen von SYSDTA oder aus einer Datei. Die Steueroptionen für den KDCDEF-Lauf (siehe OPTION-Anweisung, Seite 419) werden von KDCDEF nur dann verarbeitet, wenn sie von SYSDTA gelesen werden. Alle anderen Steueranweisungen für KDCDEF können sowohl von SYSDTA als auch aus SAM- oder ISAM-Dateien gelesen werden.

Die SAM- oder ISAM-Dateien können Sie wie folgt als Eingabequellen zuweisen:

- Eingabedatei über das BS2000-Kommando ASSIGN-SYSDTA zuweisen:

```
/ASSIGN-SYSDTA TO-FILE=inputsource
/START-EXECUTABLE-PROGRAM FROM-FILE=*MODULE
   (LIBRARY=SYSLNK.UTM.060.UTIL,ELEMENT=KDCDEF)
```

- Eingabedateien durch KDCDEF-Steueranweisung OPTION ...,DATA= zuweisen:

```
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/START-EXECUTABLE-PROGRAM FROM-FILE=*MODULE
   (LIBRARY=SYSLNK.UTM.060.UTIL,ELEMENT=KDCDEF)
OPTION DATA=inputsource1
OPTION DATA=inputsource2
usw.
END
```

Sie können den KDCDEF-Lauf auch aus WinAdmin heraus starten. Nähere Informationen dazu finden Sie auf Seite 31 und in der Online-Hilfe zu openUTM-WinAdmin.

6.2.1.2 Unix-Systeme

Um das Tool KDCDEF zu starten und eine KDCDEF-Generierung durchzuführen, gehen Sie wie folgt vor:

1. Erweitern Sie die Umgebungsvariable PATH um das Dateiverzeichnis *utmpfad*/ex.
   In diesem Dateiverzeichnis steht das Programm *kdcdef*, mit dem das Generierungstool KDCDEF gestartet wird.

2. Erzeugen Sie mit einem ASCII-Editor eine oder mehrere Source-Datei(en) mit Steueranweisungen für die UTM-Generierung. Dabei müssen Sie die Angaben in den Abschnitten „Reihenfolge der Steueranweisungen“ auf Seite 254 und „Format der Steueranweisungen“ auf Seite 255 beachten.

3. Erzeugen Sie das Dateiverzeichnis *filebase* (Basisverzeichnis der Anwendung), in dem openUTM die KDCFILE und andere Anwendungs-spezifische Dateien hinterlegt. Geben Sie dazu folgendes Kommando ein:

   ```
   mkdir␣filebase
   ```

   Das Verzeichnis müssen Sie **vor** dem Start von KDCDEF einrichten. *filebase* ist das Dateiverzeichnis, das Sie in der MAX-Anweisung im Operanden FILEBASE= angeben.

4. Das Tool KDCDEF starten Sie mit dem Programm *kdcdef*.

   Standardmäßig liest KDCDEF die KDCDEF-Steueranweisungen von *stdin*. Aus einem Shellskript werden ausschließlich die Steuer-Optionen für den KDCDEF-Lauf eingelesen (siehe OPTION-Anweisung auf Seite 419), während die eigentlichen Generierungsanweisungen für KDCDEF aus den im Schritt 2. erzeugten Dateien gelesen werden. Diese Dateien können Sie entweder direkt beim Starten von KDCDEF angeben:

   ```
   kdcdef < definput
   ```

   oder nach dem Start mit Hilfe der KDCDEF-Anweisung OPTION:

   ```
   OPTION DATA=definput
   END
   ```

Die Meldungen und Protokolle von KDCDEF werden hierbei nach *stdout* und *stderr* geschrieben, d.h. wenn Sie die Ausgabe nicht umleiten, dann bekommen Sie alles am Bildschirm angezeigt. Sie können die Ausgabe wie folgt in Dateien umleiten (die Namen der Dateien sind frei wählbar):

```
kdcdef < definput 2>def.err 1>def.prot
```

In *def.err* werden alle UTM-Meldungen protokolliert, *def.prot* enthält das komplette Protokoll des KDCDEF-Laufs.

6.2.1.3 Windows-Systeme

W Um das Tool KDCDEF zu starten und eine KDCDEF-Generierung durchzuführen, gehen Sie wie folgt vor:

1. Erweitern Sie die Umgebungsvariable PATH um das Dateiverzeichnis *utmpfad*\ex.
   In diesem Dateiverzeichnis steht das Generierungs-Programm *kdcdef.exe* und weitere Dienstprogramme und DLLs von openUTM. Dazu gehen Sie wie folgt vor:
   - Wählen Sie im Start / Einstellungen / Systemsteuerung / System und klicken Sie Umgebung an.
   - Tragen Sie für die Variable PATH den oben genannten Pfad ein und drücken Sie auf den Button „Setzen“.
2. Erzeugen Sie mit einem ASCII-Editor wie z.B. dem NOTEPAD eine oder mehrere Source-Datei(en) mit Steueranweisungen für die openUTM-Generierung. Dabei müssen Sie die Angaben in den Abschnitten „Reihenfolge der Steueranweisungen“ auf Seite 254 und „Format der Steueranweisungen“ auf Seite 255 beachten.
3. Erzeugen Sie das Dateiverzeichnis *filebase* (Projektverzeichnis), in dem openUTM die KDCFILE und andere Anwendungs-spezifische Dateien hinterlegt. Das Verzeichnis müssen Sie **vor** dem Start von KDCDEF einrichten. *filebase* ist das Dateiverzeichnis, das Sie in der MAX-Anweisung im Operanden FILEBASE= angeben.
4. Starten Sie jetzt das Tool KDCDEF. Öffnen Sie dazu mit Start / Programme / Eingabeaufforderung ein Fenster für den Kommandomodus. KDCDEF liest die KDCDEF-Steueranweisungen standardmäßig von *stdin*, d.h. direkt aus der Eingabeaufforderung heraus. Damit KDCDEF die Steueranweisungen aus einer Datei (z.B. *definput.txt*) liest, geben Sie Folgendes an:

   ```
   kdcdef < definput.txt
   ```

   oder starten Sie KDCDEF mit `kdcdef` und übergeben die Datei mit Hilfe der KDCDEF-Anweisung OPTION:

   ```
   OPTION DATA=definput.txt
   ```

Die Meldungen und Protokolle von KDCDEF werden hierbei nach *stdout* und *stderr* geschrieben, d.h. wenn Sie die Ausgabe nicht umleiten, dann bekommen Sie alles am Bildschirm angezeigt. Sie können die Ausgabe wie folgt in Dateien umleiten (die Namen der Dateien sind frei wählbar):

```
kdcdef < definput.txt 2>def.err 1>def.prot
```

In *def.err* werden alle UTM-Meldungen protokolliert, *def.prot* enthält das komplette Protokoll des KDCDEF-Laufs.

6.2.2 Reihenfolge der Steueranweisungen

Die Steueranweisungen können bis auf die im Folgenden aufgelisteten Ausnahmen in beliebiger Reihenfolge eingegeben werden. Alle Steueranweisungen außer END und UTMD können Sie mehrfach eingeben.

- Die END-Anweisung schließt als letzte Anweisung alle Steueranweisungen ab.
- Bei der OPTION-Anweisung gilt immer der zuletzt angegebene Wert des Parameters.
- Die Reihenfolge der AREA-Anweisungen muss mit der Reihenfolge übereinstimmen, mit der die Areas in der Parameterliste angegeben und in den Teilprogrammen deklariert werden (z.B. in der LINKAGE-Section in COBOL). Siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“.
- Die Reihenfolge der EXIT-Anweisungen mit USAGE=START bzw. USAGE=SHUT legt die Reihenfolge fest, in der die Programme der Event-Exits START und SHUT beim Start bzw. beim Beenden der Anwendung durchlaufen werden.
- Das Master-LTERM eines LTERM-Bündels muss vor den Slave-LTERMs dieses LTERM-Bündels generiert werden.
- Das Primary-LTERM einer LTERM-Gruppe muss vor den Alias-LTERMs dieser LTERM-Gruppe generiert werden.

B B
- Die Anweisung DEFAULT bezieht sich ausschließlich auf die nach ihr eingegebenen Steueranweisungen.

B B B B
- Die Reihenfolge der LOAD-MODULE-Anweisungen bestimmt die Reihenfolge, in der die Lademodule nachgeladen werden. Siehe dazu die LOAD-MODULE-Anweisung auf Seite 336 und das openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“.

X/W X/W
- Die Reihenfolge der SHARED-OBJECT-Anweisungen bestimmt die Reihenfolge, in der die Shared Objects/DLLs nachgeladen werden.

6.2.3 Format der Steueranweisungen

Die KDCDEF-Steueranweisungen (mit Ausnahme der DEFAULT-Anweisung in BS2000/OSD) haben das folgende Format:

control-statement␣*operand1*,*operand2*,...

- *control-statement* darf ab Spalte 1 oder später beginnen.
- Zwischen *control-statement* und den Operanden muss mindestens ein Leerzeichen stehen.
- Jede Zeile einer Steueranweisung darf maximal 240 Zeichen lang sein. Bei Verwendung von Fortsetzungszeilen darf die Steueranweisung maximal 3096 Zeichen lang sein (siehe „Fortsetzungszeilen in Steueranweisungen“ auf Seite 255).
- Kommentare können Sie mit der Anweisung REMARK oder mit einem Stern (*) in der ersten Spalte erzeugen.
- Die Anweisung EJECT bewirkt einen Seitenvorschub im Protokoll. Die EJECT-Zeile wird nicht protokolliert.

6.2.4 Fortsetzungszeilen in Steueranweisungen

Eine Steueranweisung für das Generierungstool KDCDEF kann aus einer oder mehreren Zeilen bestehen. Als Fortsetzungszeichen gilt der Bindestrich (-) oder der Gegenschrägstrich (\; Backslash). D.h. ist das letzte Zeichen einer Zeile, das kein Leerzeichen ist, ein Bindestrich oder ein Gegenschrägstrich, interpretiert KDCDEF die folgende Zeile als zur begonnenen Anweisung zugehörig. Die Folgezeile kann ab Spalte 1 oder später beginnen.

Jede Steueranweisung (außer Kommentarzeilen) kann maximal 3096 Zeichen lang sein, Fortsetzungszeichen und Leerzeichen hinter dem Fortsetzungszeichen werden dabei nicht mitgerechnet.

Kommentarzeilen sind immer einzeilig, d.h. jede Kommentarzeile muss durch REMARK oder * in der ersten Spalte der Zeile gekennzeichnet sein.

6.2.5 Syntax- und Plausibilitätsprüfung

KDCDEF führt für alle eingegebenen Steueranweisungen Syntax- und Plausibilitätsprüfungen durch. Erkennt KDCDEF dabei keine schwerwiegenden Fehler, dann erzeugt KDCDEF die KDCFILE und/oder den Sourcecode für die ROOT-Tabellen, je nachdem was Sie in OPTION angegeben haben.

Für UTM-Cluster-Anwendungen werden gegebenenfalls auch die Cluster-Konfigurationsdatei und die Cluster-User-Datei erstellt.

KDCDEF führt die Plausibilitätsprüfungen immer für alle Steueranweisungen durch. Soll in einem KDCDEF-Lauf z.B. nur eine ROOT-Tabellen-Source erzeugt werden, dann prüft KDCDEF auch die Steueranweisungen, die sich nur auf die KDCFILE auswirken.

Aus diesem Grund sollten Sie jeden KDCDEF-Lauf mit der gesamten Generierungsinformation durchführen, unabhängig davon, ob nur der Sourcecode für die ROOT-Tabellen oder nur die KDCFILE erzeugt werden soll.

Durch diese vollständige Plausibilitätsprüfung können Inkonsistenzen bei der Erzeugung von ROOT-Tabellenmodul und KDCFILE, die sonst erst beim Start der Anwendung entdeckt würden, frühzeitig erkannt werden. Folgefehler werden vermieden.

6.2.6 KDCDEF-Protokollierung

Die KDCDEF-Protokollierung können Sie zur besseren Lesbarkeit wie folgt strukturieren:

- Sie können Kommentare ins KDCDEF-Protokoll einfügen:
  - KDCDEF-Steueranweisung "*kommentar*"
    Der hinter einer KDCDEF-Steueranweisung eingefügte Kommentar darf kein weiteres Anführungszeichen enthalten.
  - * *kommentar* oder **REM**ARK *kommentar*
    Ein * oder REMARK in Spalte 1 erzeugt eine Kommentarzeile mit Zeilennummer.
- Sie können vor KDCDEF-Steueranweisungen mit .*marke* Marken setzen. Beachten Sie, dass vor *marke* ein Punkt stehen muss. *marke* darf max. acht alphanumerische Zeichen lang sein, muss aber mit einem Buchstaben beginnen.
- Die Anweisung EJECT bewirkt einen Seitenvorschub im Protokoll. Die EJECT-Zeile wird nicht protokolliert.

6.2.7 Format und Eindeutigkeit der Objektnamen

Bei der Konfigurierung von Objekten der Anwendung müssen Sie den Objekten Namen zuordnen, mit denen openUTM bzw. ein Benutzer die Objekte ansprechen kann. Bei der Vergabe der Namen müssen Sie Folgendes beachten:

– Sie dürfen keine reservierten Namen benutzen.
– Der Name eines Objektes muss innerhalb seiner Objektklasse eindeutig sein.
– Die Namen dürfen die vorgeschriebene Maximallänge nicht überschreiten und dürfen nur bestimmte Zeichen enthalten (Format).

6.2.7.1 Reservierte Namen

Damit die Vergabe von reservierten Namen nicht zu unerwartetem, undefiniertem Verhalten der UTM-Anwendung führt, beachten Sie bitte die folgenden Hinweise:

- Namen, die mit KDC beginnen, sind für die Transaktionscodes der Event-Services, der Administrationskommandos (KDCADM), der Dead Letter Queue und der SAT-Administration (BS2000/OSD) reserviert und sollten auch nur für diese Objekte verwendet werden.
  B B Eine Ausnahme bilden lediglich Lademodule einer UTM-Anwendung unter BS2000/OSD.
- B B In BS2000/OSD sollten Teilprogrammnamen nicht mit Präfixen beginnen, die für Laufzeitsysteme verwendet werden, wie IT, IC etc..
- X/W X/W X/W In Unix- und Windows-Systemen sollten die Namen für UTM-Objekte nicht mit KC, x, ITS oder mF beginnen. Externe Namen (z.B. Teilprogrammnamen) sollten nicht mit ’t_’, ’a_’, ’o_’ oder ’s_’ beginnen. Sie sind für CMX (t_) bzw. OSS (a_, o_, s_) reserviert.

6.2.7.2 Format der Namen

Für Namen, die Sie in KDCDEF-Steueranweisungen eingeben, müssen Sie die folgenden Konventionen beachten:

- Der Basisname der KDCFILE (MAX...,KDCFILE=) muss den Regeln für Dateinamen des Betriebssystems entsprechen, in dem die Anwendung ablaufen soll (siehe MAX-Anweisung auf Seite 368).

- Namen von LTERM-Partnern, Clients und Druckern, sowie Transaktionscodes und TAC-Queues etc. können max. 8 Zeichen lang sein, wobei die folgenden Zeichen in beliebiger Kombination erlaubt sind:
  - A,B,C,...,Z
  - 0,1,...,9
  - #, @, $

X/W
X/W

Unter Unix- und Windows-Systemen dürfen die Namen auch Kleinbuchstaben (a,b,c,...,z) enthalten. Zwischen Klein- und Großbuchstaben wird unterschieden.

- Programmnamen, die in der PROGRAM-Anweisung als Entry-/Objektname angegeben werden, dürfen maximal 32 Zeichen lang sein. Dies gilt auch für Programmnamen in TAC PROGRAM= und EXIT PROGRAM=.

  Bei Programmnamen sind folgende Zeichen erlaubt:

  - A,B,C,...,Z
  - 0,1,...,9
  - #, @, $

  Werden weitere Sonderzeichen angegeben, dann muss der Programmname in Hochkommata eingeschlossen werden, siehe unten.

- Zusätzliche Sonderzeichen in Namen:
  - Namen von Programmen oder Passwörtern können auch andere Sonderzeichen enthalten wie z.B. "_" (Unterstrich) und "-" (Bindestrich), wenn dies die jeweilige Systemumgebung erlaubt.

B
B

  - Die Lademodul-Namen in BS2000/OSD (LOAD-MODULE) dürfen zusätzlich noch die Zeichen "." (Punkt) und "-" (Bindestrich) enthalten.
  - Namen, die andere Sonderzeichen enthalten (Programmnamen, Passwörter, ...), müssen Sie in Hochkommata einschließen.
    Siehe dazu z.B. auch USER...,PROTECT-PW= auf Seite 546.

- Ausnahmen zur Namenslänge:
  - Presentation- und Session-Selektoren in der ACCESS-POINT- und OSI-CON-Anweisung können bis zu 16 Zeichen lang sein.
  - Programmnamen können bis zu 32 Zeichen lang sein.

B
B

  - In BS2000/OSD können Lademodul-Namen bis zu 32 Zeichen lang sein, Namen von Common Memory Pools (Mpools) dürfen bis zu 50 Zeichen lang sein.

6.2.7.3 Anzahl der Namen

Für jede der folgenden Steueranweisungen wird jeweils ein Name erzeugt:

ACCESS-POINT	MUX (BS2000/OSD)
BCAMAPPL	OSI-CON
CON	OSI-LPAP
EDIT	PROGRAM
KSET	PTERM
LOAD-MODULE (BS2000/OSD)	SHARED-OBJECT(Unix-, Windows-Systeme)
LPAP	TAC
LSES	TLS
LTAC	TPOOL
LTERM	USER
MASTER-OSI-LPAP	ULS
MASTER-LU61-LPAP	

Für die Steueranweisungen CLUSTER, LTERM, MUX und TPOOL wird in den folgenden Fällen mehr als ein Name generiert:

- Für die CLUSTER-Anweisung werden insgesamt 98 Namen erzeugt: 32 User, 32 PTERM, 32 LTERM, 1 BCAMAPPL, 1 TPOOL.
- Wird die Anwendung ohne USER generiert, dann werden pro LTERM-Anweisung zwei Namen erzeugt.
- Für eine zu einer PTERM-Anweisung mit PTYPE=APPLI, SOCKET, UPIC-R oder UPIC-L gehörenden LTERM-Anweisung werden zwei Namen erzeugt, wenn der zu diesem LTERM gehörende implizite (Verbindungs-)USER nicht durch eine explizite USER-Anweisung generiert wird.
- Für jede TPOOL-Anweisung werden doppelt so viele Namen erzeugt, wie im Operanden NUMBER= der TPOOL-Anweisung angegeben wurden. Für eine TPOOL-Anweisung mit PTYPE=APPLI, SOCKET, UPIC-R werden **dreimal** so viele Namen erzeugt, wie im Operanden NUMBER= angegeben.

B
- Für jede MUX-Anweisung werden zwei Namen erzeugt.

Darüber hinaus werden bei der Generierung bis zu sechs weitere Namen erzeugt, die openUTM für Event Services benötigt (KDCSGNTC, KDCBADTC, KDCMSGTC, KDCMSGUS, KDCMSGLT, KDCAPLKS). Die ersten drei Namen können zusätzlich in einer TAC-Anweisung spezifiziert werden. Die letzten drei Namen dürfen nicht angegeben werden.

Werden für eine UTM-Anwendung XATMI-Teilprogramme generiert, d.h. wird mindestens in einer TAC-Anweisung API=(XOPEN,XATMI) gesetzt, dann werden von openUTM zusätzlich ein TAC-Eintrag mit dem Namen KDCTXCOM und ein PROGRAM-Eintrag mit dem Namen KDCTXRLB erzeugt.

Für die Dead Letter Queue wird bei der Generierung der Name KDCDLETQ erzeugt. Die Eigenschaften dieser TAC-Queue können auch durch eine eigene TAC-Anweisung definiert werden.

Maximalwerte für Namen

Für die Anzahl der per KDCDEF-Steueranweisung erzeugbaren Namen gelten die Maximalwerte, die Sie der folgenden Tabelle entnehmen können. Wird diese Anzahl überschritten, führt dies zum Abbruch der Generierung.

	Gruppe der KDCDEF-Steueranweisungen	maximale Anzahl generierter Namen
	#USER + #APPLI + #LSES + #OSI-ACTIVE-ASSOCIATIONS + (2 * #TASKS) + 1	≤ 500 000
B	#PTERM + #CON + TPOOLNR + #OSI-ASSOCIATIONS + #MUX[1]	≤ 500 000
B	#LTERM + #LPAP + TPOOLNR + #OSI-LPAP + #TASKS + #MUX[1] + 1	≤ 500 000
X/W	#PTERM + #CON + TPOOLNR + #OSI-ASSOCIATIONS	≤ 500 000
X/W	#LTERM + #LPAP + TPOOLNR + #OSI-LPAP + #TASKS + 1	≤ 500 000
	#PROGRAM	≤ 32 000
	#TAC + 4	≤ 32 000
	#LSES	≤ 65 000
	#CON	≤ 65 000
	#KSET + 1	≤ 32 000
	#LTAC	≤ 32 000
	#MUX[1]	≤ 9 999
	Summe aller übrigen Namen + 2	≤ 32 767

B [1] MUX wird nur in BS2000/OSD unterstützt.

Erläuterung der Platzhalter:

#*statement*	Anzahl der durch diese KDCDEF-Anweisung generierten Namen
#APPLI	Anzahl der PTERM-Anweisungen plus die Werte TPOOLNR der TPOOL-Anweisungen mit PTYPE=APPLI/SOCKET/UPIC-R und UPIC-L (UPIC-L nur in Unix- und Windows-Systemen).

Wenn eine CLUSTER-Anweisung angegeben ist, erhöht sich #APPLI um 32.

B #MUX Anzahl aller generierten MUX-Anweisungen (nur in BS2000/OSD)

#OSI-ACTIVE-ASSOCIATIONS
Anzahl der aktiven parallelen OSI-Verbindungen, der generierten Werte des Operanden (OSI-CON...,ACTIVE=YES und zugehörige OSI-LPAP...,ASSOCIATIONS=*number*). Das ist die Summe aller Werte von ASSOCIATIONS über alle OSI-LPAP-Anweisungen.

#OSI-ASSOCIATIONS
Anzahl der OSI-ACTIVE-ASSOCIATIONS plus der Anzahl der inaktiven parallelen OSI-Verbindungen (OSI-CON...,ACTIVE=YES/NO und zugehörige OSI-LPAP...,ASSOCIATIONS=*number*). Das ist die Summe aller Werte von ASSOCIATIONS über alle OSI-LPAP-Anweisungen, wobei die Werte der OSI-LPAP-Anweisungen mehrfach gezählt werden, für die Ersatzverbindungen generiert werden.

TPOOLNR Summe der Operanden NUMBER= (Anzahl der LTERM-Partner pro LTERM-Pool) aller generierten TPOOL-Anweisungen.

Wenn eine CLUSTER-Anweisung angegeben ist, erhöht sich TPOOLNR um 32.

Zusätzlich ist Folgendes zu beachten:

- Die Anzahl der Namen für #PROGRAM, #TAC, #LTERM, #PTERM, #USER, #KSET, #LTAC, #CON und #LSES setzt sich aus statisch generierten Namen und reservierten Namen für dynamisch eintragbare Objekte zusammen.
- Die Namen von MASTER-LU61-LPAP-Anweisungen müssen bei #LPAP mitgezählt werden.
- Die Namen von MASTER-OSI-LPAP-Anweisungen müssen bei #OSI-LPAP mitgezählt werden.
- Wenn die Anwendung ohne USER-Anweisungen generiert wurde, muss in der ersten Bedingung #USER durch #LTERM + TPOOLNR ersetzt werden.
- Es können maximal 100 ULS- und 100 TLS-Blöcke generiert werden.
- Die Anzahl der generierten Benutzerkennungen (#USER) plus die Anzahl der für die Vorgangskellerung vorgesehenen Einträge (festgelegt in MAX NRCONV=) ist auf maximal 500 000 beschränkt.
- Die Anzahl der generierten Benutzerkennungen (#USER) plus die Anzahl der für die Vorgangskellerung (MAX NRCONV=) vorgesehenen Einträge plus die Anzahl der maximal möglichen parallelen Asynchron-Vorgänge (festgelegt in

MAX ASYNTASKS = (...,*service_number*)) plus die Anzahl der für Anmelde-Vorgänge (SIGNON CONCURRENT-TERMINAL-SIGNON) vorgesehenen Einträge ist auf maximal 665 000 beschränkt.

6.2.7.4 Eindeutigkeit der Namen und Adressen

Die Objekte einer UTM-Anwendung sind in gemeinsamen Namensräumen zusammengefasst, die für bestimmte Typen von Objekten definiert sind. Innerhalb einer Namensklasse müssen die Adressen und Namen der Objekte der zulässigen Typen eindeutig sein. Ein Name bzw. eine Adresse darf innerhalb einer Namensklasse nur einmal vergeben werden. Es gibt drei Namensklassen:

1. Namensklasse

- LTERM-Partner (Anweisung LTERM *ltermname*)
- LTERM-Partner, die openUTM für die LTERM-Pools erzeugt (Anweisung TPOOL...,LTERM=*ltermprefix*, NUMBER=*number*)
- Transaktionscodes und TAC-Queues (Anweisung TAC *tacname*)
- LPAP- bzw. OSI-LPAP-Partner für die Server-Server-Kommunikation (Anweisungen OSI-LPAP *osi_lpap_name* und LPAP *lpapname*)

2. Namensklasse

- Benutzerkennungen (Anweisung USER *username*)
- Sessions für die verteilte Verarbeitung über LU6.1 (Anweisung LSES *sessionname*)
- Verbindungen und Associations für die verteilte Verarbeitung über OSI TP (Anweisung OSI-LPAP..., ASSOCIATION-NAMES=, ASSOCIATIONS=)

3. Namensklasse

- Clients und Drucker (PTERM-Anweisung)
  Clients sind hierbei Terminals, UPIC-Clients, Transportsystem-Anwendungen (DCAM-, PDN-, CMX-, Socket-Anwendungen) und UTM-Partner-Anwendungen, die bei der Kommunikation kein höheres Kommunikationsprotokoll nutzen (LU6.1, OSI TP).
- Name der Partner-Anwendung bei der verteilten Verarbeitung über das Protokoll LU6.1 (CON-Anweisung)
- Name der Partner-Anwendung bei der verteilten Verarbeitung über das Protokoll OSI TP (OSI-CON-Anweisung)

B
- Multiplexanschlüsse einer UTM-Anwendung unter BS2000/OSD (MUX-Anweisung)

Die aufgeführten Objekte sind Kommunikationspartner der UTM-Anwendung. Sie bzw. die Verbindungen zu ihnen müssen für openUTM eindeutig identifizierbar sein. Deshalb wird jeder Kommunikationspartner durch openUTM über ein Namenstripel identifiziert. Die Namenstripel müssen in der UTM-Anwendung eindeutig sein und setzen sich aus den folgenden Komponenten zusammen:

- Name des Kommunikationspartners
  Er wird angegeben in *ptermname* der PTERM-Anweisung, *remote_appliname* der CON-Anweisung, TRANSPORT-SELECTOR= der OSI-CON-Anweisung und *name* der MUX-Anweisung.

B
B
  In BS2000/OSD muss der BCAM-Name des Kommunikationspartners angegeben werden.

- Name des Rechners, auf dem sich der Kommunikationspartner befindet.
  Er wird angegeben in den Operanden PRONAM= der PTERM-, CON- und MUX-Anweisung und im Operanden NETWORK-SELECTOR= der OSI-CON-Anweisung.

- Name der lokalen Anwendung, über den die Verbindung zum Kommunikationspartner aufgebaut wird. Er wird angegeben in den Operanden BCAMAPPL= der PTERM-, MUX- und CON-Anweisung und im Operanden LOCAL-ACCESS-POINT= der OSI-CON-Anweisung.

6.2.8 Ergebnis des KDCDEF-Laufs

Das Generierungstool KDCDEF erzeugt abhängig von den Angaben bei der Generierung folgende Objekte:

- BS2000/OSD:

B – die KDCFILE mit der Hauptdatei *filebase*.KDCA und bei doppelter Dateiführung das Duplikat *filebase*.KDCB
B – die ROOT-Tabellen-Source
B – den Pagepool *filebase*.P*nn*A mit ggf. Duplikat *filebase*.P*nn*B
B – den Wiederanlaufbereich *filebase*.R*nn*A mit ggf. Duplikat *filebase*.R*nn*B

X/W
- Unix-, Windows-Systeme:

X/W – die Hauptdatei KDCA im Dateiverzeichnis *filebase* und bei doppelter Dateiführung das Duplikat KDCB ebenfalls im Basisverzeichnis *filebase*.
X/W – die ROOT-Tabellen-Source als C/C++-Source
X/W – den Pagepool P*nn*A mit ggf. Duplikat P*nn*B im Dateiverzeichnis *filebase*
X/W – den Wiederanlaufbereich R*nn*A mit ggf. Duplikat R*nn*B im Basisverzeichnis *filebase*

- zusätzlich, wenn eine UTM-Cluster-Anwendung generiert wird, siehe auch Kapitel „Hinweise zur Generierung einer UTM-Cluster-Anwendung“ auf Seite 63:
  - eine Cluster-Konfigurationsdatei
  - eine Cluster-User-Datei

Der Aufbau der KDCFILE wird ausführlich im Abschnitt „Die KDCFILE“ auf Seite 44 beschrieben.

KDCDEF meldet nach SYSOUT (BS2000/OSD) bzw. nach *stderr* (Unix-, Windows-Systeme), ob die KDCFILE erfolgreich erzeugt wurde und zusätzlich die Größe des KAA (KDC Application Area), den die Anwendung belegt. Außerdem schreibt KDCDEF ein Protokoll mit den Steueranweisungen und evtl. Fehlermeldungen nach SYSLST (BS2000/OSD) bzw. nach *stdout* (Unix-, Windows-Systeme).

B **Hinweis zu KDCDEF in BS2000/OSD**

B Beendet sich das Generierungstool KDCDEF in BS2000/OSD auf Grund eines Fehlers
B nicht normal, dann wird von KDCDEF der Prozess-Schalter 3 gesetzt (wie bei allen UTM-
B Tools im Fehlerfall), und es werden keine Dateien angelegt. Eine Ausnahme bilden evtl.
B durch die Anweisung CREATE-CONTROL-STATEMENTS erzeugte Dateien.

6.3 Inverser KDCDEF

Damit bei einer Neugenerierung Ihrer Anwendung alle Änderungen der Konfiguration, die Sie während des Betriebs der Anwendung dynamisch vorgenommen haben, nicht verloren gehen, stellt Ihnen openUTM die Funktion „inverser KDCDEF“ zur Verfügung. Mit dem inversen KDCDEF können Sie aus den Konfigurationsdaten in der aktuellen KDCFILE Steueranweisungen für das Generierungstool KDCDEF erzeugen.

Der inverse KDCDEF erzeugt Steueranweisungen für Objekttypen, für die das dynamische Eintragen und Löschen möglich ist:

- **USER-Anweisungen**

  für alle aktuell in der Anwendung existierenden Benutzerkennungen. Der inverse KDCDEF erzeugt dabei keine USER-Anweisungen für Benutzerkennungen, die openUTM intern für LTERM-Partner von Clients des Typs UPIC-R, APPLI und SOCKET erzeugt hat.

- **LTERM-Anweisungen**

  für alle LTERM-Partner der Anwendung, die nicht zu einem LTERM-Pool oder einem Multiplexanschluss (BS2000/OSD) gehören.

- **PTERM-Anweisungen**

  für alle Clients und Drucker, die in der Konfiguration eingetragen sind. Für Clients, die sich über einen LTERM-Pool an die Anwendung anschließen oder zu einem Multiplexanschluss gehören, werden keine PTERM-Anweisungen erzeugt.

- **PROGRAM-Anweisungen**

  für alle Teilprogramme und VORGANG-Exits, die aktuell in der Konfiguration der Anwendung enthalten sind.

- **TAC-Anweisungen**

  für alle Transaktionscodes und TAC-Queues der Anwendung.

- **KSET-Anweisungen**

  für alle Keysets der Anwendung.

- **CON-Anweisungen**

  für alle LU6.1-Verbindungen der Anwendung.

- **LSES-Anweisungen**

  für alle LU6.1-Session-Namen der Anwendung.

- **LTAC-Anweisungen**

  für alle lokalen Transaktionscodes für VTV-Partner-Anwendungen.

Beim inversen KDCDEF werden auch Steueranweisungen für Objekte der oben aufgeführten Objekttypen erzeugt, die bei einer vorherigen KDCDEF-Generierung statisch erzeugt wurden. Alle Modifikationen, die für diese Objekte während des Anwendungslaufs dynamisch eingetragen wurden, werden beim inversen KDCDEF berücksichtigt.

Der inverse KDCDEF erzeugt **keine** Steueranweisungen für andere Objekte als die der oben aufgeführten Objekttypen, noch erzeugt der inverse KDCDEF Steueranweisungen für weitere Komponenten der Anwendung sowie Anwendungsparameter.

Für Objekte, die dynamisch aus der Konfiguration der Anwendung gelöscht wurden, erzeugt der inverse KDCDEF **keine** Steueranweisungen. Diese Objekte sind somit nach der Neugenerierung endgültig aus der Konfiguration gelöscht. Sie belegen keine Tabellenplätze mehr und die Namen der Objekte können wieder verwendet werden.

Zu dynamisch gelöschten Objekten überträgt das Tool KDCUPD nach der Neugenerierung mit KDCDEF auch keine Anwendungsdaten aus der alten in die neue KDCFILE. Auch dann nicht, wenn in der neuen KDCDEF-Generierung ein Objekt mit Namen und Objekttyp eines gelöschten Objektes vorhanden ist. Insbesondere werden von KDCUPD keine Asynchron-Aufträge übertragen, die über inzwischen gelöschte LTERM-Partner oder Benutzerkennungen erzeugt wurden.

Die vom inversen KDCDEF erzeugten USER-Anweisungen enthalten keine Passwörter. Für Benutzerkennungen, die mit Passwort generiert sind, erzeugt der inverse KDCDEF USER-Anweisungen der Form:

```
USER username, PASS=*RANDOM,....
```

Nach dem Abschluss des KDCDEF-Laufs, wenn die neue KDCFILE erzeugt wurde, müssen Sie die Passwörter der Benutzerkennungen mit dem Tool KDCUPD in die neue KDCFILE übertragen. Das ist auch bei einer UTM-F-Anwendung möglich. Siehe dazu auch das Kapitel „Das Tool KDCUPD - KDCFILE aktualisieren“ auf Seite 585.

Für UTM-Cluster-Anwendungen ist eine Übertragung der Passwörter mit KDCUPD i.A. nicht erforderlich. In UTM-Cluster-Anwendungen sind die aktuellen Passwörter in der Cluster-User-Datei abgelegt – nicht in der KDCFILE.

Nur wenn eine neue Cluster-User-Datei erzeugt wurde und Sie die Passwörter vom letzten Anwendungslauf erhalten möchten, müssen Sie die Passwörter mit KDCUPD übertragen.

Um dabei sicherzustellen, dass die KDCFILE die aktuellen Passwörter enthält, muss vor dem Beenden der Anwendung einmal die aktuelle Information zu allen Usern gelesen werden, z.B. mittels WinAdmin.

6.3.1 Inversen KDCDEF starten

Sie können den inversen KDCDEF „online“ oder „offline“ starten.

„Online“ heißt, Sie starten den inversen KDCDEF-Lauf im laufenden Betrieb mit dem Aufruf KC_CREATE_STATEMENTS über die Programmschnittstelle zur Administration. Informationen dazu finden Sie im openUTM-Handbuch „Anwendungen administrieren“.

„Offline“ können Sie den inversen KDCDEF nur starten, wenn das Anwendungsprogramm nicht läuft, d.h. außerhalb des Anwendungsbetriebs. Da der inverse KDCDEF die Daten aus der KDCFILE liest, muss sichergestellt sein, dass die Daten der KDCFILE während des inversen KDCDEF-Laufs nicht modifiziert werden.

Sie starten den inversen KDCDEF offline, indem Sie das Generierungstool KDCDEF aufrufen und die Steueranweisung **CREATE-CONTROL-STATEMENTS** absetzen. Siehe dazu auch CREATE-CONTROL-STATEMENTS-Anweisung auf Seite 314.

Sie können den inversen KDCDEF so starten, dass KDCDEF-Steueranweisungen entweder für alle zulässigen Objekttypen erzeugt werden oder nur für bestimmte Objekttypen, die in den Objektgruppen CON, DEVICE, KSET, LSES, LTAC, PROGRAM und USER zusammengefasst sind.

- `CREATE-CONTROL-STATEMENTS *ALL`

  Für alle Objekte der Objekttypen TAC, PROGRAM, PTERM, LTERM, USER, KSET, LTAC, CON und LSES werden KDCDEF-Steueranweisungen erzeugt.

- `CREATE-CONTROL-STATEMENTS DEVICE`

  Es werden LTERM- und PTERM-Anweisungen für LTERM-Partner, Clients und Drucker erzeugt.

- `CREATE-CONTROL-STATEMENTS PROGRAM`

  Es werden PROGRAM- und TAC-Anweisungen für Teilprogramme, VORGANG-Exits und Transaktionscodes erzeugt.

- `CREATE-CONTROL-STATEMENTS USER`

  Es werden USER-Anweisungen für Benutzerkennungen erzeugt.

- `CREATE-CONTROL-STATEMENTS KSET`

  Es werden KSET-Anweisungen für Keysets erzeugt.

- `CREATE-CONTROL-STATEMENTS LTAC`

  Es werden LTAC-Anweisungen für Transaktionscodes erzeugt, über die Service-Programme in Partner-Anwendungen gestartet werden.

- `CREATE-CONTROL-STATEMENTS CON`

  Es werden CON-Anweisungen für Transportverbindungen zu entfernten LU6.1-Anwendungen erzeugt.

- `CREATE-CONTROL-STATEMENTS LSES`

  Es werden LSES-Anweisungen für die Vergabe von LU6.1-Sessionnamen erzeugt.

6.3.2 Ergebnis des inversen KDCDEF

Beim inversen KDCDEF können Sie festlegen, dass entweder alle Steueranweisungen in eine Datei oder die Steueranweisungen einer Objektgruppe jeweils in eine eigene Datei geschrieben werden. Den Namen der Datei(en), die der inverse KDCDEF erzeugt, legen Sie beim Start des inversen KDCDEF mit der Anweisung CREATE-CONTROL-STATEMENTS fest. Existiert keine Datei dieses Namens, dann wird sie automatisch angelegt. Existiert eine Datei dieses Namens, dann können Sie festlegen, ob sie überschrieben oder fortgeschrieben werden soll.

Die Anweisung CREATE-CONTROL-STATEMENTS wirkt unmittelbar. Sie können deshalb im selben KDCDEF-Lauf direkt nach der CREATE-CONTROL-STATEMENTS-Anweisung die OPTION-Anweisung absetzen, in der Sie die vom inversen KDCDEF erzeugten Dateien an KDCDEF übergeben:

```
:
CREATE-CONTROL-STATEMENTS *ALL, TO-FILE=control_statements_file
                               , MODE=CREATE,FROM-FILE=kdcfile
OPTION DATA=control_statements_file
:
:
END
```

Die folgende Grafik zeigt, wie Sie die Dateien, die der inverse KDCDEF erzeugt, direkt als Input-Dateien an KDCDEF übergeben. Sie können die Dateien aber auch editieren, d.h. vor dem KDCDEF-Lauf modifizieren und später, bei einer Neugenerierung, an KDCDEF übergeben. Jede erzeugte Input-Datei weisen Sie KDCDEF mit der Steueranweisung OPTION DATA=*control_statements_file* zu.

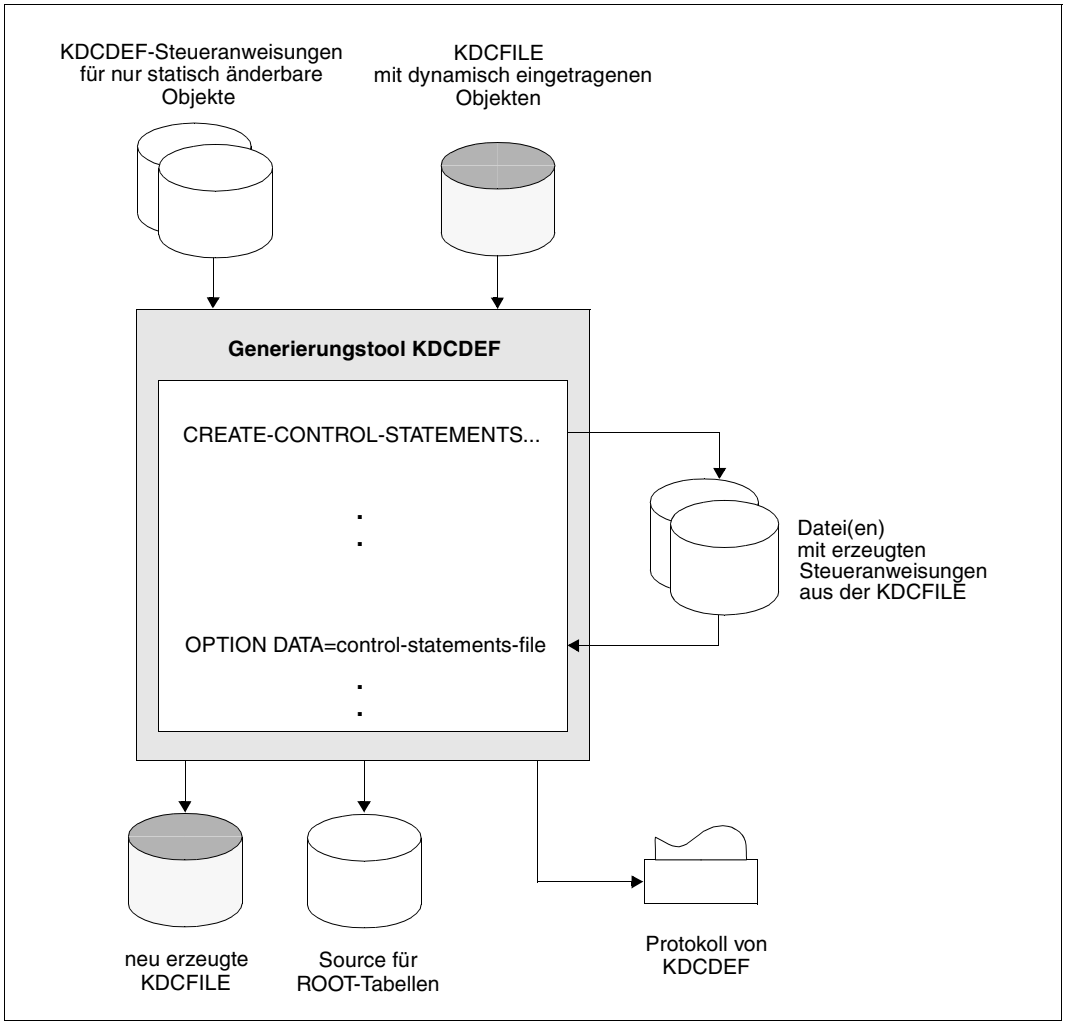


Bild 18: KDCDEF-Lauf mit inversem KDCDEF

6.3.3 Erzeugen von KDCDEF-Steueranweisungen bei Versionsübergängen

Damit der inverse KDCDEF Informationen aus der KDCFILE lesen kann, muss die KDCFILE mit derselben openUTM-Version erzeugt worden sein, zu der auch das beim inversen KDCDEF-Lauf verwendete Generierungstool KDCDEF gehört.

Bei einem Übergang auf eine neue openUTM-Version müssen Sie die KDCDEF-Steueranweisungen zunächst in der Vorgängerversion erzeugen, d.h. Sie müssen den inversen KDCDEF der Vorgängerversion starten. Die von ihm erzeugten Dateien können Sie als Input-Dateien für den KDCDEF der neuen openUTM-Version verwenden.

6.4 Empfehlungen für die Neugenerierung einer Anwendung

Beim Betrieb einer UTM-Anwendung ist es manchmal unumgänglich, eine Anwendung neu zu generieren.

Für UTM-Cluster-Anwendungen gibt es einerseits Änderungen, die mit einer Neugenerierung bei laufender UTM-Cluster-Anwendung gemacht werden können, andererseits Änderungen, die man nur durchführen kann, wenn die UTM-Cluster-Anwendung vollständig beendet wurde.

Eine Liste der Änderungen, die vor dem Start mit der neuen KDCFILE ein vollständiges Beenden der UTM-Cluster-Anwendung erfordern, entnehmen Sie folgenden Handbüchern:

- openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“
- openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter Unix- und Windows-Systemen“

Mögliche Gründe für einen erneuten KDCDEF-Lauf sind:

- Die bei der Generierung festgelegten Maximalwerte müssen angepasst werden.
- Für die verteilte Verarbeitung über OSI TP und LU6.1 müssen möglicherweise neue Objekte erzeugt werden, weil sich beispielsweise der Server-Verbund bei der verteilten Verarbeitung vergrößern soll.
  Für die verteilte Verarbeitung über LU6.1 ist ein KDCDEF-Lauf nur dann erforderlich, wenn neue LPAP-Objekte eingefügt werden müssen. Objekte vom Typ CON, LSES und LTAC lassen sich dagegen auch durch dynamische Administration erzeugen (vorausgesetzt es wurden ausreichend Tabellenplätze mit der RESERVE-Anweisung freigehalten).

- Neue Lademodule (BS2000/OSD), Shared Objects (Unix-Systeme) oder DLLs (Windows-Systeme) müssen in das Anwendungsprogramm eingefügt werden.
- Die reservierten Tabellenplätze für das dynamische Eintragen von Objekten in die Konfiguration sind belegt. Die Tabellen müssen erweitert werden oder zum Löschen vorgemerkte Objekte müssen endgültig gelöscht werden, um die Tabellenplätze und die Namen der Objekte zur Wiederverwendung freizugeben.

Die Ausfallzeit Ihrer Anwendung, die eine Neugenerierung mit sich bringt, können Sie verringern, wenn Sie die folgenden Empfehlungen beachten:

- Teilen Sie bereits bei der Erstgenerierung Ihrer Anwendung die KDCDEF-Steueranweisungen auf verschiedene Dateien auf, und zwar getrennt nach nur statisch zu generierenden und dynamisch eintragbaren Objekten, die Sie KDCDEF als Input-Dateien mit der Anweisung OPTION DATA= zur Verfügung stellen.
- Schreiben Sie die Steueranweisungen USER, LTERM, PTERM, PROGRAM, TAC, CON, KSET, LSES und LTAC jeweils nach Objektgruppen getrennt in Dateien, damit Sie bei einer Neugenerierung der Anwendung diese Dateien einfach durch die vom inversen KDCDEF erzeugten Dateien (DEVICE, PROGRAM, USER, CON, KSET, LSES und LTAC) ersetzen können. Siehe dazu auch die Anweisung CREATE-CONTROL-STATEMENTS auf Seite 314.

Vor einer Neugenerierung der Anwendung und vor dem inversen KDCDEF-Lauf empfiehlt es sich alle Objekte, die in der neuen Konfiguration nicht mehr enthalten sein sollen, dynamisch mit dem Aufruf KC_DELETE_OBJECT zu löschen. Siehe dazu das openUTM-Handbuch „Anwendungen administrieren“.

In UTM-Cluster-Anwendungen müssen Objekte, die sich dynamisch administrieren lassen, immer administrativ gelöscht werden. Ein Löschen nur in der KDCDEF-Source führt zu Inkonsistenzen in den einzelnen Knoten-Anwendungen der UTM-Cluster-Anwendung.

Das dynamische Löschen hat gegenüber dem manuellen Löschen der zugehörigen Steueranweisungen aus der Input-Datei für den KDCDEF-Lauf folgende Vorteile:

- Wird bei der Neugenerierung ein Objekt aus der Input-Datei manuell gelöscht und im selben Generierungslauf ein Objekt gleichen Typs und Namens aber mit anderen Eigenschaften neu definiert, dann kann das im Anschluss laufende Tool KDCUPD die Objekte nicht als zwei verschiedene Objekte erkennen und überträgt die Daten des gelöschten Objekts für das neue Objekt in die KDCFILE. Dieses unerwünschte Verhalten tritt nicht auf, wenn Sie das Objekt zuvor dynamisch löschen und dann bei der Neugenerierung ein Objekt gleichen Typs und Namens erzeugen. In diesem Fall erkennt KDCUPD, dass es sich um verschiedene Objekte handelt und überträgt die Daten des alten Objekts nicht in die neue KDCFILE.

- Das manuelle Löschen von KDCDEF-Anweisungen aus der KDCDEF-Eingabedatei ist unkomfortabel und fehleranfällig. Es muss beim Löschen auf Abhängigkeiten zwischen den Objekten und damit zwischen den KDCDEF-Anweisungen geachtet werden. Werden Abhängigkeiten übersehen, muss der KDCDEF-Lauf wiederholt werden. Die Ausfallzeit wird damit vergrößert.
- Die Abläufe bei der Neugenerierung können automatisiert werden. Sie können in einer Prozedur den inversen KDCDEF aufrufen und die von ihm erzeugten Dateien dem KDCDEF direkt zur Verfügung stellen. Im Anschluss daran kann die Prozedur das Tool KDCUPD aufrufen. Durch diesen vollautomatischen prozeduralen Ablauf wird die Ausfallzeit bei einer Neugenerierung minimiert.

Um unerwünschte Folgen zu vermeiden, beachten Sie bitte beim dynamischen Löschen, ob für Objekte, die in der laufenden Anwendung dynamisch gelöscht bzw. gesperrt werden, nicht noch wartende Aufträge etc. existieren.

6.5 Steueranweisungen von KDCDEF

ABSTRACT-SYNTAX - Abstrakte Syntax definieren

Die Steueranweisung ABSTRACT-SYNTAX wird nur benötigt, wenn Sie für die Kommunikation über das OSI-TP-Protokoll einen eigenen Application Context definieren wollen (siehe Anweisung APPLICATION-CONTEXT Seite 283).

ABSTRACT-SYNTAX definiert für eine Abstrakte Syntax einen lokalen Namen und weist diesem einen Object-Identifier und die Transfersyntax zu, die zur Übertragung der Benutzerdaten ausgewählt wurde. openUTM generiert automatisch die Abstrakten Syntaxen CCR, UDT, XATMI und UTMSEC. Diese müssen deshalb nicht explizit mit der ABSTRACT-SYNTAX-Anweisung generiert werden. Es können maximal 50 Abstrakte Syntaxen generiert werden, inklusive der von openUTM implizit generierten Syntaxen.

```
ABSTRACT-SYNTAX␣ abstract_syntax_name
                   ,OBJECT-IDENTIFIER=object_identifier
                 [ ,TRANSFER-SYNTAX=transfer_syntax_name ]
```

abstract_syntax_name
: Ein max. 8 Zeichen langer Name, der lokal für eine Abstrakte Syntax vergeben wird. Innerhalb einer UTM-Anwendung muss jede Abstrakte Syntax einen eigenen Namen haben.

  abstract_syntax_name muss beim MGET/MPUT bzw. FGET/FPUT angegeben werden, wenn Daten dieser Abstrakten Syntax gesendet oder empfangen werden sollen.

OBJECT-IDENTIFIER=object_identifier
: Object-Identifier der Abstrakten Syntax, der wie folgt angegeben wird:

  object_identifier=(number1,number2, ... ,number10)

  number ist dabei eine positive ganze Zahl im Bereich von 0 bis 67108863. Für *object_identifier* werden in Klammern und durch Komma getrennt mindestens zwei bis maximal zehn Zahlen angegeben. Die Anzahl und Position der Zahlen ist relevant.

  Anstelle der Zahl kann auch der dieser Zahl zugeordnete symbolische Name angegeben werden. Der Tabelle auf Seite 92 können Sie entnehmen, welchen Wert *number* an dieser Position annehmen darf.

  object_identifier muss innerhalb der UTM-Anwendung eindeutig sein, d.h. es dürfen keine weiteren Abstrakten Syntaxen mit demselben Object-Identifier generiert werden.

TRANSFER-SYNTAX=transfer_syntax_name

Name einer mit der Steueranweisung TRANSFER-SYNTAX vergebenen Transfer-Syntax.

Standard: BER (Basic Encoding Rules)

openUTM generiert automatisch die Abstrakten Syntaxen CCR, UDT, XATMI und UTM-SEC, die wie folgt definiert sind:

Generierung von „CCR“:

```
ABSTRACT-SYNTAX␣CCR,                                          -
               OBJECT-IDENTIFIER=(2, 7, 2, 1, 2),             -
               TRANSFER-SYNTAX=BER
```

Symbolische Beschreibung des Object-Identifiers:

```
(joint-iso-ccitt, ccr, abstract-syntax, apdus, version2)
```

Generierung von „UDT“:

```
ABSTRACT-SYNTAX␣UDT,                                          -
               OBJECT-IDENTIFIER=(1, 0, 10026, 6, 1, 1),      -
               TRANSFER-SYNTAX=BER
```

Symbolische Beschreibung des Object-Identifiers:

```
(iso, standard, tp, udt, generic-abstract-syntax, version)
```

Generierung von „XATMI“:

```
ABSTRACT-SYNTAX␣XATMI,                                        -
               OBJECT-IDENTIFIER=(1, 2, 826, 0, 1050, 4, 1, 0), -
               TRANSFER-SYNTAX=BER
```

Symbolische Beschreibung des Object-Identifiers:

```
(iso, national-member-body, bsi, disc, xopen, xatmi, apdus-abstract-
syntax, version1)
```

Generierung von „UTMSEC“:

```
ABSTRACT-SYNTAX␣UTMSEC,                                                -
               OBJECT-IDENTIFIER=(1, 3, 0012, 2, 1107, 1, 6, 1, 2, 0),  -
               TRANSFER-SYNTAX=BER
```

Symbolische Beschreibung des Object-Identifiers:

```
(iso, identified-organisation, icd-ecma, member-company, siemens-units,
sni, transaction-processing, utm-security, abstract-syntax, version)
```

ACCESS-POINT - OSI TP-Zugriffspunkt einrichten

Die Steueranweisung ACCESS-POINT wird nur für die Kommunikation über das OSI TP-Protokoll benötigt. Sie definiert einen lokalen Zugriffspunkt zu den Diensten von OSI TP.

Wenn mehr als eine ACCESS-POINT-Anweisung pro Anwendung geschrieben wird, gibt KDCDEF die Warnung K492 aus.

Mit den Angaben, die in der ACCESS-POINT-Anweisung gemacht werden, kann eine Partner-Anwendung die lokale Anwendung adressieren. Im einzelnen legen Sie in der ACCESS-POINT-Anweisung folgende Parameter für einen Zugriffspunkt fest:

- Adresse des Zugriffspunktes innerhalb des lokalen Systems

  Die Adresse des Zugriffspunktes besteht aus den Komponenten Presentation-Selektor, Session-Selektor und Transport-Selektor.

  Die Adressangaben müssen mit den Kommunikationspartnern abgestimmt werden. Die Angabe für TRANSPORT-SELECTOR ist in jedem Fall Pflicht.

X/W *Unix- und Windows-Systeme:*
X/W Auf Unix- und Windows-Systemen besteht die Adresse des Zugriffspunktes zusätzlich
X/W aus den Komponenten LISTENER-PORT, T-PROT und TSEL-FORMAT.
X/W Siehe dazu Abschnitt „Adressinformationen für das Transportsystem CMX bereitstellen (Unix- und Windows-Systeme)“ auf Seite 105.

- Application Entity Qualifier
  Als weitere Adresskomponente können Sie einen Application Entity Qualifier (AEQ) vergeben. Der AEQ bildet zusammen mit dem Application Process Title (APT), den Sie in der UTMD-Anweisung definieren, den Application Entity Title (AET).
  Ein AET ist ein global eindeutiger Name für eine Anwendung (= Application Entity) innerhalb der OSI TP-Umgebung. Bei transaktionsgesicherter Verarbeitung benötigt die Partner-Anwendung den AET der lokalen UTM-Anwendung für den Verbindungsaufbau. Ebenso benötigt die lokale Anwendung den AET der Partner-Anwendung. Er muss in der OSI-LPAP-Steueranweisung, die die Partner-Anwendung beschreibt, angegeben werden.
  Die Angabe eines Transport-Selektors für den Zugriffspunkt ist weiterhin Pflicht.

X/W
X/W
X/W
X/W
X/W
X/W
- Listener-ID (Unix-, Windows-Systeme)
  Unter Unix- und Windows-Systemen wird dem Zugriffspunkt eine Listener-ID zugeordnet, wenn die Netzanbindung der Anwendung multi-threaded erfolgt. Netzverbindungen, die von demselben Netzprozess verwaltet werden sollen, können so zusammengefasst werden. Unter Windows-Systemen erfolgt die Netzanbindung immer multi-threaded.

Jeder ACCESS-POINT wird beim Anwendungsstart, falls möglich, beim Transportsystem angemeldet und erst beim Anwendungsende wieder abgemeldet.

```
ACCESS-POINT␣ access_point_name

              [ ,APPLICATION-ENTITY-QUALIFIER=aequalifier ]

                ,PRESENTATION-SELECTOR={ *NONE |
                                         (C'c' [,STD | EBCDIC | ASCII ]) |
                                         X'x' }

                ,SESSION-SELECTOR={ *NONE |
                                    (C'c' [, STD | EBCDIC | ASCII ]) |
                                    X'x' }

                ,TRANSPORT-SELECTOR=C'c'
```

zusätzliche Operanden in Unix-, Windows-Systemen

```
X/W           [ ,LISTENER-ID=number ]
X/W           [ ,LISTENER-PORT=number ]
X/W           [ ,T-PROT=( RFC1006 ) ]
X/W           [ ,TSEL-FORMAT={ T | E | A } ]
```

access_point_name

Name des OSI TP-Zugriffspunktes. Innerhalb der lokalen UTM-Anwendung wird der OSI TP-Zugriffspunkt durch diesen Namen identifiziert.

access_point_name kann max. 8 Zeichen lang sein. *access_point_name* muss in der lokalen UTM-Anwendung eindeutig sein.

APPLICATION-**E**NTITY-**Q**UALIFIER=aequalifier

Adresskomponente des Application Entity Title (AET). Der AET wird dann benötigt, wenn mit Transaktionssicherung (Commit Functional Unit) gearbeitet wird, oder aber ein heterogener Partner einen AET für den Verbindungsaufbau erwartet.

Die Angabe eines Application Entity Qualifiers (AEQ) ist nur erlaubt, wenn in der UTMD-Anweisung auch ein Application Process Title (APT) für die Anwendung angegeben wird.

Einem APT hingegen muss nicht zwingend ein AEQ zugeordnet werden. Wird jedoch **kein** AEQ definiert, dann hat der Zugriffspunkt auch keinen Application Entity Title (AET), d.h. über den so generierten Zugriffspunkt kann nicht mit Transaktionssicherung (Commit Functional Unit) gearbeitet werden.

Enthält der Application Context eines OSI-LPAP-Partners, der über diesen Zugriffspunkt arbeitet (OSI-CON-Anweisung), die CCR-Syntax, dann ist die Angabe eines Application Entity Qualifiers Pflicht.

Für *aequalifier* wird eine positive ganze Zahl angegeben. *aequalifier* muss innerhalb der Anwendung eindeutig sein, d.h. *aequalifier*=zahl1 darf in keiner anderen ACCESS-POINT-Anweisung als AEQ angegeben werden.

Minimalwert: 1
Maximalwert: 67108 863 (2^{26}-1)

X/W LISTENER-ID=number

X/W ordnet dem Zugriffspunkt eine Listener-ID zu, als Verwaltungsinformation bei multi-threaded Netzanbindung.

X/W Listener-IDs können Sie für Zugriffspunkte und Anwendungsnamen angeben. Siehe dazu auch die BCAMAPPL-Anweisung auf Seite 289.

X/W Bei multi-threaded Netzanbindungen (MAX...,NET-ACCESS= MULTI-THREADED) können in einem Netzprozess mehrere Verbindungen verwaltet werden. Alle Verbindungen mit derselben Listener-ID werden dabei innerhalb desselben Netzprozesses verwaltet. Mit Hilfe der Listener-IDs können Sie die Verwaltung der Netzanbindungen auf verschiedene Netzprozesse verteilen.

X/W Wenn Sie keine Listener-ID angeben, dann nimmt openUTM als Listener-ID den Wert 0 an und fasst alle Verbindungen ohne Listener-ID in einem Netzprozess zusammen.

X/W Standardwert: 0
Minimalwert: 0
Maximalwert: 32767

X/W BCAMAPPL-Namen, die für die Kommunikation über die Socket-Schnittstelle (native TCP/IP) erzeugt wurden, benutzen eigene Netzprozesse. Ihre Listener-IDs bilden einen eigenen Nummernraum, d.h. auch wenn Sie dieselbe Listener-ID wie dieser Zugriffspunkt haben, werden Sie in einem anderen Netzprozess verwaltet.

X/W LISTENER-PORT=number

X/W Portnummer des Zugriffspunktes für den Aufbau von TCP/IP-Verbindungen

X/W Erlaubte Werte: 102 und 1025 - 32767

X/W Standard: 0 (d.h. keine Portnummer)

X/W Bei OPTION CHECK-RFC1006=YES muss bei LISTENER-PORT eine Portnummer angegeben werden.

PRESENTATION**-SEL**ECTOR=

bezeichnet den Presentation-Selektor der Adresse des OSI TP-Zugriffspunktes.

*NONE Die Adresse des OSI TP-Zugriffspunktes enthält keinen Presentation-Selektor.

C'c' Der Presentation-Selektor wird als Characterstring (c) angegeben. Der für *c* angegebene Wert kann maximal 16 Zeichen lang sein. Groß- und Kleinbuchstaben werden unterschieden.

Bei einem Characterstring können Sie wählen, in welchem Code die Zeichen interpretiert werden.

STD Die Zeichen werden in Maschinen-spezifischer Codierung interpretiert (BS2000/OSD = EBCDIC; Unix- und Windows-Systeme = ASCII).

Standard: STD

EBCDIC
Die Zeichen werden in EBCDIC-Codierung interpretiert.

ASCII Die Zeichen werden in ASCII-Codierung interpretiert.

X'x' Der Presentation-Selektor wird als Hexadezimalzahl (x) angegeben. Für *x* dürfen maximal 32 Hexadezimalziffern (≙ 16 Byte) angegeben werden. Es muss eine gerade Anzahl von Hexadezimalziffern angegeben werden.

SESSION**-SEL**ECTOR=

bezeichnet den Session-Selektor der Adresse des OSI TP-Zugriffspunktes.

*NONE Die Adresse des OSI TP-Zugriffspunktes enthält keinen Session-Selektor.

C'c' Der Session-Selektor wird als Characterstring (c) angegeben. Der für *c* angegebene Wert kann maximal 16 Zeichen lang sein. Groß- und Kleinbuchstaben werden unterschieden.

Bei einem Characterstring können Sie wählen, in welchem Code die Zeichen interpretiert werden.

STD Die Zeichen werden in Maschinen-spezifischer Codierung interpretiert (BS2000/OSD = EBCDIC; Unix- und Windows-Systeme = ASCII).

Standard: STD

EBCDIC
Die Zeichen werden in EBCDIC-Codierung interpretiert.

ASCII Die Zeichen werden in ASCII-Codierung interpretiert.

X'x' Der Session-Selektor wird als Hexadezimalzahl (x) angegeben. Für *x* dürfen maximal 32 Hexadezimalziffern (≙ 16 Byte) angegeben werden. Es muss eine gerade Anzahl von Hexadezimalziffern angegeben werden.

TRANSPORT-**SEL**ECTOR=C'c'

Transport-Komponente der Adresse des OSI TP-Zugriffspunktes

Die Angabe von T-SEL=C'c' ist Pflicht.

Für T-SEL= können Sie max. 8 abdruckbare Zeichen angeben. Zulässige Zeichen sind Großbuchstaben, Ziffern und die Zeichen $, # und @. Bindestriche im Namen sind nicht erlaubt. Das erste Zeichen des Namens muss ein Großbuchstabe sein.

Der in T-SEL angegebene Name muss in der lokalen UTM-Anwendung eindeutig sein. Er darf nicht übereinstimmen mit dem primären Anwendungsnamen in MAX APPLINAME, einem BCAMAPPL-Namen oder mit einem T-Selektor einer anderen ACCESS-POINT-Anweisung.

B *BS2000/OSD*:

B Für T-SEL= ist der lokale BCAM-Anwendungsname anzugeben. Der T-Selektor muss im lokalen System eindeutig sein.
B

X/W *Unix-, Windows-Systeme:*

X/W Sie müssen T-SEL mit dem T-Selektoren der OSI TP-Partner abstimmen.
X/W Ist der Partner z.B. eine UTM-Anwendung, dann muss die Angabe bei
X/W T-SEL mit dem T-Selektor der OSI-CON-Anweisung beim Partner übereinstimmen.
X/W

X/W T-PROT= Adressformate der T-Selektoren des Zugriffspunktes.

X/W Weitere Informationen finden Sie im CMX-Benutzerhandbuch oder in der
X/W Online-Hilfe zu PCMX.

X/W RFC1006 Adressformat RFC1006, ISO-Transportprotokoll über TCP/IP und Konver-
X/W genzprotokoll RFC1006.

X/W Standard: RFC1006

X/W TSEL-FORMAT=

X/W X/W X/W Formatindikator der T-Selektoren des Zugriffspunktes (Operand TRANSPORT-SELECTOR).

X/W X/W X/W Der Formatindikator gibt die Codierung des T-Selektors im Transportprotokoll an. Nähere Informationen finden Sie im CMX-Benutzerhandbuch bzw. in der Online-Hilfe zu PCMX.

X/W T TRANSDATA-Format (die Codierung erfolgt in EBCDIC)

X/W E EBCDIC-Zeichenformat

X/W A ASCII-Zeichenformat

X/W X/W X/W X/W Standard:
T wenn der Zeichenvorrat des T-Selektors dem TRANSDATA-Format entspricht
E sonst

X/W Es wird empfohlen, explizit einen Wert für TSEL-FORMAT anzugeben.

ACCOUNT - Accounting-Funktionen festlegen

Mit der Steueranweisung ACCOUNT definieren Sie:

- ob beim Start der UTM-Anwendung die Abrechnungs- oder Kalkulationsphase des UTM-Accountings eingeschaltet werden soll,
- wann ein Abrechnungssatz geschrieben wird,
- wie die verbrauchten Betriebsmittel in der Abrechnungsphase gewichtet werden.

Wird die Steueranweisung ACCOUNT nicht angegeben, so hat das die gleiche Wirkung wie ACCOUNT ACC=NO.

Das UTM-Accounting kann auch per Administration ein- und ausgeschaltet werden, auch dann, wenn bei der KDCDEF-Generierung keine ACCOUNT-Anweisung abgesetzt wird. Es gelten dann die Standardwerte.

Sie dürfen die ACCOUNT-Anweisung innerhalb eines KDCDEF-Laufs nur einmal angeben.

Die UTM-Accounting-Funktionen und der Aufbau der Accounting-Sätze, die openUTM schreibt, sind im openUTM-Handbuch „Einsatz von openUTM-Anwendungen“ beschrieben.

```
ACCOUNT␣      ACC={ YES | NO | CALC }
              [ ,CPUUNIT=cpuunit ]
              [ ,IOUNIT=iounit ]
              [ ,MAXUNIT=maxunit ]
              [ ,OUTUNIT=outunit ]
```

ACC= gibt an, welche UTM-Accounting-Funktionen durchgeführt werden sollen. ACC ist Pflichtoperand.

- YES: Nach dem Start der Anwendung soll openUTM die Abrechnungsphase des UTM-Accounting einschalten.
- NO: Nach dem Start der Anwendung sind die Accounting-Funktionen nicht eingeschaltet.
  Sie können im laufenden Betrieb mit dem Administrationskommando KDCAPPL ..., ACC=ON oder über die Programmschnittstelle zur Administration eingeschaltet werden (siehe openUTM-Handbuch „Anwendungen administrieren“).
- CALC: Nach dem Start der Anwendung soll openUTM die Kalkulationsphase einschalten.

CPUUNIT=cpuunit

gibt das Gewicht an, mit dem eine CPU-Sekunde in der Abrechnungsphase des UTM-Accountings bewertet wird. Teile einer CPU-Sekunde werden anteilig berechnet.
Die Angabe ist nur ganzzahlig möglich.

Standardwert: 0
Minimalwert: 0
Maximalwert: 32767

IOUNIT=iounit

gibt das Gewicht an, mit dem 100 Platten-IOs in der Abrechnungsphase bewertet werden.

Weniger als 100 Ein- und Ausgaben werden anteilig berechnet.
Die Angabe ist nur ganzzahlig möglich.

Standardwert: 0
Minimalwert: 0
Maximalwert: 32767

X/W X/W X/W

Unix-, Windows-Systeme
Dieser Operand ist ohne Bedeutung, da diese Betriebssysteme keine Informationen über Platten-I/Os zur Verfügung stellen.

MAXUNIT=maxunit

gibt die Anzahl der Verrechnungseinheiten an, bei denen openUTM einen Abrechnungssatz für einen Benutzer (USER) schreiben soll.
Die Angabe ist nur ganzzahlig möglich.

Standardwert:
99 999 999 (=10^8-1)
d.h. im Normalfall wird nur beim Verbindungsabbau ein Abrechnungssatz geschrieben.

Minimalwert: 1
Maximalwert: 99 999 999 (=10^8- 1)

OUTUNIT=outunit

gibt das Gewicht an, mit dem ein Druckauftrag (FPUT NE) in der Abrechnungsphase bewertet wird. Die Angabe ist nur ganzzahlig möglich.

Standardwert: 0
Minimalwert: 0
Maximalwert: 4095

APPLICATION-CONTEXT - Application Context definieren

Die Steueranweisung APPLICATION-CONTEXT wird nur für die Kommunikation über das OSI TP-Protokoll benötigt. Sie müssen die Anweisung APPLICATION-CONTEXT nur absetzen, wenn Sie einen zusätzlichen Application Context definieren wollen.

Mit der APPLICATION-CONTEXT-Anweisung definieren Sie den Application Context, der bei der Kommunikation über das OSI TP-Protokoll verwendet wird. Der Application Context legt die Regeln der Datenübertragung zwischen den Kommunikationspartnern fest. Er gibt an, wie die Benutzerdaten für die Übertragung codiert werden und in welcher Form die Daten übertragen werden. Der Application Context muss mit dem Partner abgestimmt werden.

Die APPLICATION-CONTEXT-Anweisung definiert einen lokalen Namen für einen Application Context und weist diesem einen Object-Identifier und die Abstrakten Syntaxen zu, die zu diesem Application Context gehören.

openUTM generiert die Standard-Application Contexts UDTAC, UDTDISAC, XATMIAC, UDTCCR, UDTSEC und XATMICCR.

```
APPLICATION-CONTEXT␣  application_context_name

                      ,OBJECT-IDENTIFIER=object_identifier

                      ,ABSTRACT-SYNTAX={ abstract_syntax_name |
                                         (abstract_syntax_name,...) }
```

application_context_name

Ein max. 8 Zeichen langer Name, der lokal für einen Application Context vergeben wird.

Innerhalb einer UTM-Anwendung muss jeder *application_context_name* eindeutig sein.

OBJECT-IDENTIFIER=object_identifier

Object-Identifier des Application Context, der wie folgt angegeben wird:

object_identifier=(number1,number2, ... ,number10)

number ist dabei eine positive ganze Zahl im Bereich von 0 bis 67108863. Für *object_identifier* werden in Klammern und durch Komma getrennt mindestens zwei bis maximal zehn Zahlen angegeben. Die Anzahl und Position der Zahlen sind relevant.
Anstelle der Zahl kann auch der dieser Zahl zugeordnete symbolische Name angegeben werden. Der Tabelle auf Seite 92 können Sie entnehmen, welchen Wert *number* an dieser Position annehmen darf.

object_identifier muss innerhalb der UTM-Anwendung eindeutig sein, d.h. es dürfen keine weiteren Application Contexts mit demselben Object-Identifier generiert werden.

ABSTRACT-SYNTAX=
Abstrakte Syntax für die Übertragung der Benutzerdaten; wird dem Application Context *application_context_name* zugeordnet.

abstract_syntax_name
Name einer mit der ABSTRACT-SYNTAX-Anweisung definierten Abstrakten Syntax

(abstract_syntax_name, ..., abstract_syntax_name)
Liste von maximal 9 Abstrakten Syntaxen, die durch Kommata getrennt werden. Jede Abstrakte Syntax *abstract_syntax_name* muss zuvor mit einer ABSTRACT-SYNTAX-Anweisung definiert worden sein.

Die UTM-Standardsyntaxen CCR, UDT, XATMI und UTMSEC müssen nicht explizit generiert werden.

Um mit verteilter Transaktionverarbeitung zu arbeiten, muss ein Application Context ausgewählt werden, der die abstrakte Syntax CCR enthält.

Wenn beim APRO-Aufruf SignOn-Daten mit übergeben werden sollen, muss ein Application Context ausgewählt werden, der die abstrakte Syntax UTMSEC enthält.

Wenn beide Partner die XATMI-Schnittstelle verwenden, muss ein Application Context gewählt werden, der die abstrakte Syntax XATMI enthält.

openUTM generiert automatisch 6 Application Contexts mit den Namen UDTAC, UDTDISAC, XATMIAC, UDTCCR, UDTSEC und XATMICCR.

Die Application Contexts werden folgendermaßen definiert:

Generierung von „UDTAC“:

```
APPLICATION-CONTEXT␣UDTAC,                                    -
               OBJECT-IDENTIFIER=(1, 0, 10026, 6, 2),         -
               ABSTRACT-SYNTAX=UDT
```

Symbolische Beschreibung des Object-Identifiers:

```
(iso, standard, tp, udt, application-context)
```

Generierung von „UDTDISAC“:

```
APPLICATION-CONTEXT␣UDTDISAC,                                    -
               OBJECT-IDENTIFIER=(1, 0, 10026, 6, 2, 1),         -
               ABSTRACT-SYNTAX=UDT
```

Symbolische Beschreibung des Object-Identifiers:

```
(iso, standard, tp, udt, application-context, with-tp)
```

Generierung von „XATMIAC“:

```
APPLICATION-CONTEXT␣XATMIAC,                                     -
               OBJECT-IDENTIFIER=(1, 2, 826, 0, 1050, 4, 2, 1),  -
               ABSTRACT-SYNTAX=(XATMI)
```

Symbolische Beschreibung des Object-Identifiers:

```
(iso, national-member-body, bsi, disc, xopen, xatmi, application-context,
atpll-21-31)
```

Generierung von „UDTCCR“:

```
APPLICATION-CONTEXT␣UDTCCR,                                -
               OBJECT-IDENTIFIER=(1, 0, 10026,´6, 2),      -
               ABSTRACT-SYNTAX=(UDT, CCR)
```

Symbolische Beschreibung des Object-Identifiers:

```
(iso, standard, tp, udt, application-context)
```

Generierung von „UDTSEC“:

```
APPLICATION-CONTEXT␣UDTSEC,                                                -
               OBJECT-IDENTIFIER=(1, 3, 0012, 2, 1107, 1, 6, 1, 3, 0),     -
               ABSTRACT-SYNTAX=(UDT, UTMSEC, CCR)
```

Symbolische Beschreibung des Object-Identifiers:

```
(iso, identified-organisation, icd-ecma, member-company, siemens-units,
sni, transaction-processing, utm-security, application-context, version)
```

Generierung von „XATMICCR“:

```
APPLICATION-CONTEXT␣XATMICCR,                                    -
               OBJECT-IDENTIFIER=(1, 2, 826, 0, 1050, 4, 2, 1),  -
               ABSTRACT-SYNTAX=(XATMI, CCR)
```

Symbolische Beschreibung des Object-Identifiers:

```
(iso, national-member-body, bsi, disc, xopen, xatmi, application-context,
atpll-21-31)
```

AREA - Zusätzliche Datenbereiche (Areas) definieren

Mit der AREA-Anweisung werden Namen, Eigenschaften und Reihenfolge zusätzlicher gemeinsam verwendbarer Datenbereiche (Areas) definiert. Die Struktur dieser Areas ist von openUTM nicht vorgegeben, sondern frei definierbar. Die Adressen solcher Areas werden den Teilprogrammen beim Programmstart als Parameter übergeben, zusammen mit den Adressen von KB und SPAB.

Alternativ zur Verwaltung von Areas durch openUTM mit Hilfe von AREA-Anweisungen haben Sie bei den meisten Programmiersprachen (insbesondere bei COBOL und C/C++) die Möglichkeit, anstelle von Areas externe Datenbereiche zu deklarieren und aus den Teilprogrammen heraus anzusprechen. Diese Möglichkeit bringt gegenüber AREAs eine Reihe von Vorteilen mit sich. Näheres hierzu finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“.

Jede Area, die von openUTM verwaltet werden soll, ist mit einer eigenen AREA-Anweisung zu definieren. Die Reihenfolge der AREA-Anweisungen gibt die Reihenfolge vor, in der die Areas in der Parameterliste angegeben werden müssen und im Teilprogramm zu deklarieren sind (z.B. unter BS2000/OSD in der LINKAGE-SECTION in COBOL). Wird die an n-ter Stelle definierte Area benötigt, so müssen sowohl in der Parameterliste als auch in der Datendeklaration alle Areas bis zu dieser angegeben bzw. deklariert werden.

Innerhalb eines Generierungslaufs können unabhängig vom Betriebssystem maximal 99 AREA-Anweisungen angegeben werden.

AREAs in Cluster-Anwendungen sind Knoten-lokal, d.h. jede Knoten-Anwendung hat ein eigenes Exemplar jeder AREA.

B **Areas in BS2000/OSD generieren**

B Areas unter BS2000/OSD können abgelegt sein:

B
- im anwendungsglobalen Common Memory Pool (für alle Anwendungen).

B B
- im anwendungslokalen Common Memory Pool (für alle Prozesse von Anwendungen, die unter derselben Benutzerkennung gestartet wurden).

B
- in nichtprivilegierten Subsystemen.

B
- im gebundenen Anwendungsprogramm.

B Für die AREA-Anweisung gilt Folgendes:

B B B
- Wenn Sie den Operanden LOAD-MODULE= angeben, müssen Sie zusätzlich eine LOAD-MODULE-Anweisung schreiben. Beachten Sie, dass kein Lademodul referenziert werden darf, das mit LOAD-MODULE ...LOAD-MODE=ONCALL generiert ist.

B B
- AREA-Anweisungen, die den Operanden LOAD-MODULE nicht enthalten, definieren Areas, die statisch zum Anwendungsprogramm dazugebunden werden.

B B
- DEFAULT-Werte für AREA werden über die Anweisung DEFAULT PROGRAM eingestellt.

B
```
AREA␣        areaname
             { [ ,LOAD-MODULE=lmodname ] |
```

areaname Name der Area. *areaname* ist alphanumerisch und kann max. 32 Zeichen lang sein. *areaname* muss ein Modul sein.

LOAD-MODULE=lmodname
lmodname kann max. 32 Zeichen lang sein.

LOAD-MODULE= bezeichnet den Namen des Lademoduls, in den das Modul gebunden ist. Das Lademodul muss mit einer LOAD-MODULE-Anweisung definiert werden. Dieses Lademodul darf nicht mit dem Operanden LOAD-MODE=ONCALL generiert werden.

X/W **Areas in Unix- und Windows-Systemen generieren**

X/W Eine Area müssen Sie als externe C/C++-Datenstrukturen selbst definieren, übersetzen
X/W und zum Teilprogramm binden.

X/W In der AREA-Anweisung können Sie festlegen, ob die Area direkt an das Teilprogramm
X/W übergeben wird, oder indirekt als Zeiger auf die Area zur Verfügung gestellt wird. Beim in-
X/W direkten Zugriff muss der Zeiger vor dem Start des ersten Teilprogramms mit der Adresse
X/W der Area versorgt werden. Die Adresse können Sie zum Übersetzungszeitpunkt setzen
X/W oder während des Anwendungslaufs, z.B. im Event-Exit START.

X/W
```
AREA␣        areaname
             [ ,ACCESS={ DIRECT | INDIRECT } ]
```

X/W areaname Name der Area. *areaname* ist alphanumerisch und kann max. 32 Zeichen
X/W lang sein. *areaname* muss ein Modul sein.

X/W ACCESS= gibt an, wie der Zugriff auf die Area erfolgt.

X/W DIRECT Die Area wird direkt als C-Datenstruktur definiert.

X/W Standard: DIRECT

X/W INDIRECT Die Area *areaname* wird als Zeiger definiert. Der Zeiger *areaname* muss mit
X/W der Adresse der Area versorgt werden. Es ist möglich, dass Sie die Adresse
X/W erst während des Anwendungslaufs setzen, z.B. können Sie im Event-Exit
X/W START den Zeiger mit der Adresse eines Shared Memory versorgen.

BCAMAPPL - Weitere Anwendungsnamen definieren

Mit der BCAMAPPL-Anweisung können Sie der UTM-Anwendung weitere Anwendungsnamen für die Client/Server-Kommunikation und die verteilte Verarbeitung über LU6.1 zuordnen. Jedem BCAMAPPL-Namen müssen Sie eine Adresse innerhalb des lokalen Systems (T-Selektoren bzw. Stationsnamen) zuordnen, damit die Anwendung vom Kommunikationspartner adressierbar ist.

Der primäre Anwendungsname der UTM-Anwendung wird in der MAX-Anweisung mit dem Operanden APPLINAME festgelegt. Für ihn ist Folgendes zu beachten:

B
B
B
B

- *BS2000/OSD:*
  Sie dürfen BCAMAPPL-Anweisungen nur für weitere BCAM-Namen der Anwendung absetzen. Der primäre Anwendungsname darf nicht in einer BCAMAPPL-Anweisung angegeben werden.

X/W
X/W
X/W
X/W

- *Unix-, Windows-Systeme:*
  Sie müssen eine BCAMAPPL-Anweisung für den primären Anwendungsnamen absetzen, wenn über ihn auch Verbindungen zu Partner-Anwendungen oder Clients aufgebaut werden sollen.

Es ist nötig, weitere Anwendungsnamen für Ihre UTM-Anwendung zu generieren, wenn:

- über LU6.1 parallele Verbindungen zu anderen Anwendungen (verteilte Verarbeitung) definiert werden sollen; in diesem Fall sind zumindest in einer der beteiligten Anwendungen zusätzliche Anwendungsnamen zu generieren.
- mit einem Kommunikationspartner über die Socket-Schnittstelle (native TCP/IP) kommuniziert werden soll.
  Für die Kommunikation über die Socket-Schnittstelle benötigen Sie eigene BCAMAPPL-Namen (mit T-PROT=SOCKET). Diese können nicht für die Kommunikation über andere Transportprotokolle verwendet werden.

B
B
B

- zu einem Partner einer UTM-Anwendung unter BS2000/OSD, der mit PTYPE=APPLI, PTYPE=UPIC-R oder als Partner einer LU6.1-Anwendung generiert ist, ein Transportprotokoll ungleich NEA verwendet werden soll,

B
B

- zu einem Partner einer UTM-Anwendung unter BS2000/OSD Multiplexverbindungen aufgebaut werden sollen,

X/W
X/W
X/W

- Sie Verbindungen über das Protokoll RFC1006 Verbindungen aufbauen wollen. In diesem Fall müssen Sie für die Kommunikation über RFC1006 einen eigenen BCAMAPPL-Namen definieren.

Die Steueranweisung BCAMAPPL kann mehrfach angegeben werden. Sie sollten jedoch nur soviele BCAMAPPL-Anweisungen, d.h. zusätzliche Anwendungsnamen wie nötig generieren, um keine unnötigen Ressourcen zu belegen.

In allen K-Meldungen, die den UTM-Anwendungsnamen enthalten, erscheint der mit der MAX-Anweisung und nicht der mit der BCAMAPPL-Anweisung festgelegte Name.

B **BCAMAPPL-Anweisung unter BS2000/OSD**

```
BCAMAPPL␣     appliname
                ,LISTENER-PORT=number  nur bei T-PROT=SOCKET erlaubt und Pflicht
              [ ,SIGNON-TAC={ *NONE | tacname } ]
              [ ,T-PROT={ NEA | ISO | RFC1006 | SOCKET } ]
```

appliname Zusätzlicher BCAM-Name der UTM-Anwendung. *appliname* kann max. 8 Zeichen lang sein. *appliname* darf nicht mit den Anwendungsnamen identisch sein, die Sie in MAX...,APPLINAME= bzw. ACCESS-POINT...,TRANSPORT-SELECTOR= angeben.

Außerdem muss sich der Name auch vom Anwendungsnamen unterscheiden, den Sie im Operanden BCAMAPPL der CLUSTER-Anweisung angegeben haben.

appliname muss im lokalen System pro Host eindeutig sein.

LISTENER-PORT=number
darf nur zusammen mit T-PROT=SOCKET angegeben werden.
In diesem Fall ist LISTENER-PORT= Pflichtoperand.

LISTENER-PORT= legt die Portnummer fest, an der openUTM auf Verbindungsaufbauwünsche von außen wartet.

Es sind alle Portnummern erlaubt.

Die Portnummer kann im lokalen System nur **einmal** zum Horchen auf Verbindungsaufbauten verwendet werden.
Beim Starten von openUTM kann es sein, dass vom System oder anderen TCP/IP-Anwendungen bereits Portnummern reserviert sind oder dass privilegierte Portnummern nicht benutzt werden dürfen. In diesem Fall wird der Start der UTM-Anwendung abgebrochen.

SIGNON-TAC =

Gibt an, ob für Verbindungen, die über den Anwendungsnamen *appliname* (=Transportsystem- Zugriffspunkt) aufgebaut werden, ein Anmelde-Vorgang gestartet werden soll oder nicht. Soll ein Anmelde-Vorgang gestartet werden, dann müssen Sie den Namen des Transaktionscodes angeben, über den der Anmelde-Vorgang gestartet wird.

*NONE

Für Verbindungen, die über den Anwendungsnamen *appliname* zu der UTM-Anwendung aufgebaut werden, soll kein Anmelde-Vorgang gestartet werden, unabhängig davon, ob der TAC KDCSGNTC generiert ist oder nicht.

tacname Name des Vorgangs-TACs, über den der Anmelde-Vorgang gestartet wird.

Der Transaktionscode *tacname* muss mit einer TAC-Anweisung generiert werden. In der TAC-Anweisung dürfen Sie für den Transaktionscode die folgenden Standardeinstellungen nicht verändern:

- API = KDCS,
- CALL = FIRST oder BOTH,
- ENCRYPTION-LEVEL = NONE,
- PGWT = NO,
- TACCLASS = 0,
- TYPE = D,
- keine Einschränkung der Zugriffsrechte, d.h.die Operanden ACCESS-LIST und LOCK dürfen nicht angegeben werden

Für UPIC-Partner wird der Anmelde-Vorgang nur dann gestartet, wenn zusätzlich in der SIGNON-Anweisung UPIC=YES generiert ist. Für UPIC-Partner wird der Anmelde-Vorgang nicht beim Aufbau der Verbindung gestartet, sondern vor dem Start einer UPIC-Conversation (siehe auch SIGNON-Anweisung, Parameter OMIT-UPIC-SIGNOFF= auf Seite 486). Für LU6.1-Partner wird kein Anmelde-Vorgang gestartet.

tacname darf keinem Programm zugewiesen sein (Operand PROGRAM der TAC-Anweisung), das in einem mit LOAD-MODE=ONCALL generierten Lademodul liegt.

Standard:

- KDCSGNTC, sofern in der Anwendung generiert (KDCSGNTC = Standard-Anmelde-Vorgang; generiert mit einer TAC-Anweisung)
- in allen anderen Fällen: *NONE

VORSICHT!
Für Kommunikationspartner, die Verbindungen zur UTM-Anwendung über den primären Anwendungsnamen aufbauen (generiert in MAX APPLINAME=), kann der Anmelde-Vorgang ausschließlich über den Transaktionscode KDCSGNTC generiert werden.

T-PROT= gibt das Transportprotokoll an, das auf den Verbindungen zu Partner-Anwendungen verwendet werden soll, die über diesen Anwendungsnamen aufgebaut werden.

NEA Es soll ein NEA-Transportprotokoll verwendet werden.

Standard: NEA

ISO Es soll ein ISO-Transportprotokoll verwendet werden.

Es hängt von der Generierung des Transportsystems ab, ob eine ISO-Transportverbindung zu dieser Anwendung aufgebaut werden kann und welches Transportprotokoll letztendlich verwendet wird. Da bei ISO-Transportverbindungen parallele Verbindungen erlaubt sind, diese jedoch von openUTM nicht unterstützt werden, akzeptiert openUTM im Contention-Fall die Verbindung des Contention Winners (CON) bzw. des Partners mit dem lexikographisch kleineren Namenspaar - *ptermname*, Prozessorname - (PTERM-Anweisung).

RFC1006 Es soll als Transportprotokoll TCP/IP mit dem Konvergenzprotokoll RFC1006 verwendet werden.

Für die Kommunikation mit openUTM unter Unix-/Windows-Systemen muss RFC1006 verwendet werden.

RFC1006 wirkt im BS2000/OSD wie T-PROT=ISO.

SOCKET Es soll native TCP/IP als Transportprotokoll verwendet werden, d.h. die Kommunikation soll über die Socket-Schnittstelle erfolgen.

Wenn Sie T-PROT=SOCKET angeben, dann müssen Sie im Operanden LISTENER-PORT eine Portnummer definieren.
Näheres dazu finden Sie auf Seite 150.

X/W **BCAMAPPL-Anweisung unter Unix-, Windows-Systemen**

X/W Unter Unix- und Windows-Systemen dienen die Operanden *appliname*, LISTENER-PORT
X/W (TCP/IP-Portnummer), T-PROT (verwendete Transportprotokolle) und TSEL-FORMAT
X/W (Formatindikator) zur Angabe der Adresse.

X/W Erfolgt die Anbindung der Anwendung ans Netz multi-threaded, dann sollten Sie eine
X/W Listener-ID definieren. Die Listener-ID wird für die Zuordnung der Netzverbindungen über
X/W den BCAMAPPL-Namen zu einem Netzprozess verwendet.

```
X/W  BCAMAPPL␣     appliname
X/W                [ ,LISTENER-ID=number ]
X/W                [ ,LISTENER-PORT=number ]
X/W                [ ,SIGNON-TAC={ *NONE | tacname } ]
X/W                [ ,T-PROT={ SOCKET | RFC1006 }]
X/W                [ ,TSEL-FORMAT={ T | E | A } ]
```

X/W appliname Zusätzlicher Name der UTM-Anwendung. *appliname* kann max. 8 Zeichen
X/W lang sein. *appliname* darf nicht mit den Anwendungsnamen identisch sein,
X/W die Sie in ACCESS-POINT...,TRANSPORT-SELECTOR= angeben.

X/W Außerdem muss sich der Name auch vom Anwendungsnamen unterschei-
X/W den, den Sie im Operanden BCAMAPPL der CLUSTER-Anweisung ange-
X/W geben haben.

X/W *appliname* muss auf dem lokalen Rechner eindeutig sein.

X/W KDCDEF erzeugt aus *appliname* einen T-Selektor für das Transportsystem.
X/W Der T-Selektor ist Bestandteil der Transportadresse der Anwendung, über
X/W die diese von Partner-Anwendungen beim Verbindungsaufbau adressiert
X/W wird.

X/W Ausnahme:
X/W Bei T-PROT=SOCKET ist *appliname* nur UTM-intern von Bedeutung, z.B. für
X/W die Administration. Der Name muss nur innerhalb der Anwendung eindeutig
X/W sein.

X/W LISTENER-ID=number

X/W X/W X/W ordnet dem Anwendungsnamen eine Listener-ID zu, als Verwaltungsinformation bei multi-threaded Netzanbindungen.

X/W X/W Listener-IDs können Sie für Anwendungsnamen und Zugriffspunkte angeben. Siehe dazu auch die ACCESS-POINT-Anweisung.

X/W X/W X/W X/W X/W X/W Bei multi-threaded Netz-Anbindungen (MAX ...,NET-ACCESS= MULTI-THREADED), können in einem Netzprozess mehrere Verbindungen von Threads verwaltet werden. Alle Verbindungen mit derselben Listener-ID werden dabei innerhalb desselben Netzprozesses verwaltet. Mit Hilfe der Listener-IDs können Sie die Verwaltung der Netzanbindungen auf verschiedene Netzprozesse verteilen.

X/W X/W X/W X/W X/W X/W BCAMAPPL-Namen mit T-PROT=SOCKET (Kommunikation über die Socket-Schnittstelle) bilden einen eigenen Nummernkreis, d.h. es werden keine BCAMAPPL-Namen, die für die Kommunikation über die Socket-Schnittstelle erzeugt wurden, mit BCAMAPPL-Namen und Zugriffspunkten für andere Transportprotokolle in einem Netzprozess zusammengefasst, auch wenn die Listener-ID gleich ist.

X/W X/W X/W Alle BCAMAPPL-Namen für die Socket-Schnittstelle mit derselben Listener-ID werden immer in einem Netzprozess zusammengefasst, auch wenn NET-ACCESS=SINGLE-THREADED generiert wurde.

X/W X/W X/W X/W X/W X/W X/W Ist die multi-threaded Netzanbindung generiert und geben Sie keine Listener-ID an, dann nimmt openUTM als Listener-ID den Wert 0 an. openUTM fasst dann alle Verbindungen ohne Listener-ID zusammen. Alle Verbindungen ohne Listener-ID, die nicht über die Socket-Schnittstelle aufgebaut werden, werden in einem Netzprozess zusammengefasst und alle Verbindungen ohne Listener-ID, die über die Socket-Schnittstelle aufgebaut werden, werden in einem anderen Netzprozess zusammengefasst.

X/W X/W X/W Standardwert: 0
Minimalwert: 0
Maximalwert: 32767

X/W LISTENER-PORT=number

X/W X/W X/W Portnummer der UTM-Anwendung für den Aufbau von TCP/IP-Verbindungen.

X/W X/W X/W Erlaubte Werte für *number*:
T-PROT=RFC1006: Portnummern 102 und 1025 bis 32767 erlaubt
T-PROT=SOCKET: Portnummern zwischen 1 und 65535 erlaubt.

X/W X/W X/W Bei T-PROT=RFC1006 und OPTION CHECK-RFC1006=YES sowie bei T-PROT=SOCKET muss bei LISTENER-PORT eine Portnummer angegeben werden.

X/W In allen anderen Fällen ist der Standardwert 0 (keine Portnummer).

X/W X/W X/W – Bei der Kommunikation über die Socket-Schnittstelle (SOCKET) kann eine Portnummer nur **einmal** pro Prozessor zum Horchen auf Verbindungsaufbauten verwendet werden.

X/W X/W X/W X/W – Wenn der Standardwert verwendet wird (Portnummer 0), wird intern mit der Portnummer gearbeitet, die CMX als Standard vergibt. Dies kann zu Konflikten führen, wenn z.B. der Port durch unterschiedliche Anwendungen genutzt wird.

X/W SIGNON-TAC =

X/W X/W X/W X/W X/W X/W Gibt an, ob für Verbindungen, die über den Anwendungsnamen *appliname* (=Transportsystem- Zugriffspunkt) aufgebaut werden, ein Anmelde-Vorgang gestartet werden soll oder nicht. Soll ein Anmelde-Vorgang gestartet werden, dann müssen Sie den Namen des Transaktionscodes angeben, über den der Anmelde-Vorgang gestartet wird.

X/W X/W X/W X/W *NONE Für Verbindungen, die über den Anwendungsnamen appliname zu der UTM-Anwendung aufgebaut werden, soll kein Anmelde-Vorgang gestartet werden, unabhängig davon, ob der TAC KDCSGNTC generiert ist oder nicht.

X/W tacname Name des Vorgangs-TACs, über den der Anmelde-Vorgang gestartet wird.

X/W X/W X/W X/W X/W X/W X/W X/W X/W X/W Der Transaktionscode tacname muss mit einer TAC-Anweisung generiert werden. In der TAC-Anweisung dürfen Sie für den Transaktionscode die folgenden Standardeinstellungen nicht verändern:
- API = KDCS,
- CALL = FIRST oder BOTH,
- ENCRYPTION-LEVEL = NONE,
- PGWT = NO,
- TACCLASS = 0,
- TYPE = D,
- keine Einschränkung der Zugriffsrechte, d.h.die Operanden ACCESS-LIST und LOCK dürfen nicht angegeben werden

X/W X/W X/W X/W X/W X/W Für UPIC-Partner wird der Anmelde-Vorgang nur dann gestartet, wenn zusätzlich in der SIGNON-Anweisung UPIC=YES generiert ist. Für UPIC-Partner wird der Anmeldevorgang nicht beim Aufbau der Verbindung gestartet, sondern vor dem Start einer UPIC-Conversation, siehe auch SIGNON-Anweisung, Parameter OMIT-UPIC-SIGNOFF auf Seite 486.
Für LU6.1-Partner wird kein Anmelde-Vorgang gestartet.

X/W Standard:
X/W – KDCSGNTC, sofern in der Anwendung generiert (KDCSGNTC = Standardanmelde-Vorgang; generiert mit einer TAC-Anweisung)
X/W – in allen anderen Fällen: *NONE

X/W
X/W **VORSICHT!**
X/W Stimmt der angegebene Anwendungsname in *appliname* mit dem pimären
X/W Anwendungsnamen (generiert in MAX APPLINAME=) überein, dann dürfen
X/W Sie für SIGNON-TAC= nur KDCSGNTC (Standardanmelde-Vorgang),
X/W *NONE oder ein Leerzeichen angeben.

X/W T-PROT= Adressformate der T-Selektoren in der Transportadresse

X/W Folgende Adressformate können Sie bei T-PROT angeben:

X/W SOCKET Die Kommunikation erfolgt über die Socket-Schnittstelle.
X/W Neben T-PROT=SOCKET, LISTENER-PORT und *appliname* müssen keine
X/W weiteren Angaben zur Adresse gemacht werden.

X/W Näheres zum Adressformat SOCKET finden Sie in Abschnitt „Adressinformationen für Clients vom Typ SOCKET bereitstellen“ auf Seite 150.

X/W RFC1006 Adressformat RFC1006

X/W Näheres zum Adressformat RFC1006 finden Sie im CMX-Benutzerhand-
X/W buch oder in der Online-Hilfe zu PCMX.

X/W Standard: RFC1006

X/W TSEL-FORMAT=
X/W Formatindikator der T-Selektoren, die aus *appliname* erzeugt werden sollen.

X/W Der Formatindikator gibt die Codierung des T-Selektors im Transportproto-
X/W koll an. Nähere Informationen finden Sie im CMX-Benutzerhandbuch bzw.
X/W in der Online-Hilfe zu PCMX.

X/W T TRANSDATA-Format (Codierung erfolgt in EBCDIC).
X/W In diesem Fall muss *appliname* genau 8 Zeichen lang sein und darf keine
X/W Kleinbuchstaben enthalten.

X/W E EBCDIC-Zeichenformat

X/W A ASCII-Zeichenformat

X/W Standard:
X/W T wenn der Zeichenvorrat von *appliname* dem TRANSDATA-Format ent-
X/W spricht
X/W E sonst

X/W Für den Betrieb über RFC1006 wird jedoch empfohlen, explizit einen Wert
X/W für TSEL-FORMAT anzugeben.

CLUSTER – Globale Eigenschaften einer UTM-Cluster-Anwendung definieren

Die CLUSTER-Anweisung dient zur Konfiguration einer UTM-Cluster-Anwendung. Die Operanden der Steueranweisung CLUSTER können auf mehrere CLUSTER-Anweisungen aufgeteilt werden.

Wenn Sie den gleichen Operand in mehreren CLUSTER-Anweisungen angeben, wird die erste Angabe als gültig angenommen. Eine Meldung wird dabei nicht ausgegeben. Jeder Pflichtoperand muss einmal angegeben werden.

Wenn eine CLUSTER-Anweisung angegeben ist, müssen Sie auch mindestens zwei CLUSTER-NODE-Anweisungen angeben. Wenn eine CLUSTER-Anweisung angegeben ist, erzeugt KDCDEF implizit einen BCAMAPPL-Eintrag mit dem in der CLUSTER-Anweisung angegebenen BCAMAPPL-Namen und Port. Desweiteren wird ein TPOOL für diesen BCAMAPPL mit 32 Einträgen erzeugt.

Wenn eine CLUSTER-Anweisung und CLUSTER-NODE-Anweisungen angegeben sind, aber OPTION GEN=CLUSTER nicht angegeben ist, werden Syntax- und Plausibilitätsprüfungen für diese Anweisungen durchgeführt und die impliziten Objekte werden angelegt. Eine Cluster-Konfigurationsdatei wird nicht erzeugt. Zur Laufzeit der Anwendung wird jedoch eine Cluster-Konfigurationsdatei erwartet.

Wenn Sie bei einer Neu-Generierung Angaben in der CLUSTER-Anweisung oder den CLUSTER-NODE-Anweisungen ändern, müssen Sie eine neue Cluster-Konfigurationsdatei erstellen (OPTION GEN=CLUSTER) und einsetzen, damit die Änderungen wirksam werden.

```
CLUSTER     CLUSTER-FILEBASE = cluster_filebase
              ,BCAMAPPL = cluster_applname
              ,LISTENER-PORT = port_number
              ,USER-FILEBASE = user_filebase
            [ ,CHECK-ALIVE-TIMER-SEC = time ]
            [ ,COMMUNICATION-REPLY-TIMER-SEC = time ]
            [ ,COMMUNICATION-RETRY-NUMBER = number ]
            [ ,EMERGENCY-CMD = command_string1 ]
            [ ,FAILURE-CMD = command_string2 ]
            [ ,FILE-LOCK-RETRY = number ]
            [ ,FILE-LOCK-TIMER-SEC = time ]
            [ ,GLOBAL-UTM-DATA = { NO | YES }
            [ ,RESTART-TIMER-SEC = time ]
            [ ,USER-RESTART = { NO | YES } ]
```

B ***zusätzlicher Operand in BS2000/OSD***

B `[ ,IMPORT-USER-LOCALES = { NO | YES }`

CLUSTER-FILEBASE=cluster_filebase

Namens-Präfix bzw. Dateiverzeichnis für die Cluster-Konfigurationsdatei und andere globale Verwaltungsdateien der UTM-Cluster-Anwendung, wie z.B. für das Administrations-Journal. Für den Ablauf werden diese Dateien auf einem Speichermedium erwartet, das von allen Knoten-Anwendungen aus zugreifbar ist.

Beim Starten einer Anwendung ist der Name der Cluster-Filebase in den Startparametern unter CLUSTER-FILEBASE anzugeben.

Pflichtoperand.

B *BS2000/OSD:*

B Die Cluster-Konfigurationsdatei wird von KDCDEF mit dem Dateinamen
B *cluster_filebase*`.UTM-C.CFG` angelegt. Für den Betrieb der UTM-Cluster-An-
B wendung kann die Datei umbenannt und/oder umkopiert werden. Dabei
B muss das Namens-Suffix `.UTM-C.CFG` erhalten bleiben. Beim Start einer An-
B wendung ist der dann gültige Filebase-Name in den Startparametern anzu-
B geben. Der Name darf bis zu 42 Zeichen lang sein und muss der Syntax

B
B
B

von Dateinamen genügen. Wenn Sie den Filebase-Name ohne Catid und Userid angeben, müssen Sie trotzdem deren Länge berücksichtigen, siehe auch Abschnitt „BS2000/OSD:“ auf Seite 45.

X/W

Unix-/Windows-Systeme:

X/W
X/W
X/W

Mit *cluster_filebase* wird das Dateiverzeichnis definiert, in dem die Konfigurationsdatei abgelegt werden soll. Das Dateiverzeichnis muss vor dem KDCDEF-Lauf eingerichtet werden.

X/W

Die Konfigurationsdatei wird mit dem Dateinamen `UTM-C.CFG` angelegt.

X/W
X/W
X/W
X/W
X/W

Für den Betrieb der UTM-Cluster-Anwendung kann die Datei in ein anderes Dateiverzeichnis umkopiert werden. Geben Sie beim Start einer Anwendung den dann gültigen Namen in den Startparametern an. Der Name darf bis zu 42 Zeichen lang sein und muss der Syntax von Dateinamen genügen.

BCAMAPPL=cluster_applname

Name des Kommunikationsendpunkts für die Cluster-interne Kommunikation.

Der hier angegebene Name muss verschieden sein von den Namen, die bei MAX APPLINAME, in anderen BCAMAPPL-Anweisungen oder in ACCESS-POINT-Anweisungen beim Operanden TRANSPORT-SELECTOR angegeben wurden. Darüber hinaus darf der hier angegebene Name nicht von anderen Anwendungen auf den Rechnern der UTM-Cluster-Anwendung als Name eines Kommunikationsendpunkts verwendet werden.

Der hier generierte Name darf nicht in anderen Anweisungen (z.B. in der PTERM-Anweisung) als BCAMAPPL-Name referenziert werden.

Der Name kann bis zu 8 Zeichen lang sein.

Pflichtoperand.

LISTENER-PORT=port_number

Portnummer für die Cluster-interne Kommunikation.

Dieser Operand legt die Portnummer fest, an der die lokale Anwendung auf Verbindungsaufbauwünsche von außen wartet. Geben Sie eine beliebige Portnummer zwischen 1025 und 65535 an.

Beachten Sie, dass die hier angegebene Portnummer nicht anderweitig auf den Rechnern des UTM-Clusters verwendet werden darf. Die Portnummer muss sich auch von den anderen von dieser Anwendung verwendeten Portnummern unterscheiden. Dies wird jedoch von KDCDEF nicht überprüft.

Pflichtoperand.

USER-FILEBASE=user_filebase

Namens-Präfix bzw. Dateiverzeichnis für die Cluster-User-Datei einer UTM-Cluster-Anwendung.

Wenn die Cluster-User-Datei neu erstellt werden soll, können Sie einen beliebigen *user_filebase* angeben.

Wenn die Cluster-User-Datei bereits existiert, müssen Sie den *user_filebase* der aktuellen Cluster-User-Datei angeben. KDCDEF wertet die existierende Cluster-User-Datei aus und erweitert sie. Dabei darf die Cluster-User-Datei zeitgleich zu dem KDCDEF-Lauf von einer laufenden UTM-Cluster-Anwendung geöffnet sein.

KDCDEF erzeugt bzw. erweitert die Cluster-User-Datei immer dann, wenn eine KDCFILE für eine UTM-Cluster-Anwendung erstellt werden soll. Wenn in einem Generierungslauf keine neue KDCFILE erstellt werden soll, greift KDCDEF nur lesend auf die Cluster-User-Datei zu, um die Konsistenz der Generierungsanweisungen zu prüfen.

Pflichtoperand.

B *BS2000/OSD:*

B Wenn die Cluster-User-Datei noch nicht existiert, wird sie mit dem Dateinamen *user_filebase*`.UTM-C.USER` angelegt. Vor dem ersten Start der UTM-Cluster-Anwendung können Sie die Datei umbenennen und/oder umkopieren. Dabei muss das Namens-Suffix `.UTM-C.USER` erhalten bleiben. Beim Start der Anwendung muss die Cluster-User-Datei unter dem Filebase-Namen der Cluster-Konfigurationsdatei katalogisiert sein.

B Der Name darf bis zu 42 Zeichen lang sein und muss der Syntax von Dateinamen genügen. Wenn Sie den Filebase-Namen ohne Catid und Userid angeben, müssen Sie trotzdem deren Länge berücksichtigen.

X/W *Unix-/Windows-Systeme:*

X/W Wenn die Cluster-User-Datei noch nicht existiert, wird sie im Dateiverzeichnis *user_filebase* angelegt. Das Dateiverzeichnis müssen Sie vor dem KDCDEF-Lauf einrichten. Die Cluster-User-Datei wird mit dem Dateinamen `UTM-C.USER` angelegt. Vor dem ersten Start der UTM-Cluster-Anwendung können Sie die Datei in ein anderes Dateiverzeichnis umkopieren. Beim Start der Anwendung muss die Cluster-User-Datei im Dateiverzeichnis der Cluster-Konfigurationsdatei vorhanden sein.

X/W Der Name darf bis zu 42 Zeichen lang sein und muss der Syntax von Dateinamen genügen.

CHECK-ALIVE-TIMER-SEC=time

Zeitabstand in Sekunden, in dem eine Knoten-Anwendung einer UTM-Cluster-Anwendung die Verfügbarkeit einer anderen Knoten-Anwendung überprüft.

Minimalwert: 30
Maximalwert: 3600
Standardwert: 600

COMMUNICATION-REPLY-TIMER-SEC=time

Zeit in Sekunden, die eine Knoten-Anwendung einer UTM-Cluster-Anwendung nach einer Nachricht an eine andere Knoten-Anwendung auf die Antwort wartet.

Bei Ausbleiben der Antwort in der hier vorgegebenen Zeit muss der Ausfall der anderen Knoten-Anwendung angenommen werden. Wenn Sie für COMMUNICATION-RETRY-NUMBER einen Wert größer Null gesetzt haben, wird von einem Ausfall der anderen Knoten-Anwendung erst nach dem Ablauf aller Wiederholungsversuche ausgegangen.

Minimalwert: 1
Maximalwert: 60
Standardwert: 10

COMMUNICATION-RETRY-NUMBER=number

Anzahl von Wiederholungen einer Kommunikation mit einer anderen Knoten-Anwendung, falls diese Knoten-Anwendung nicht innerhalb der bei COMMUNICATION-REPLY-TIMER festgesetzten Zeit antwortet. Wenn die überwachte Knoten-Anwendung auch auf keinen der Wiederholungsversuche antwortet, wird diese Knoten-Anwendung als ausgefallen gekennzeichnet.

Minimalwert: 0, d.h. keine Wiederholung nach einem Timeout
Maximalwert: 10
Standardwert: 1

EMERGENCY-CMD=command_string1

Mit diesem Operanden wird ein Kommando-String übergeben, der ein auszuführendes Kommando enthält.

Das Emergency-Kommando wird von openUTM aufgerufen, wenn eine ausgefallene Knoten-Anwendung nach Aufruf des FAILURE-Skripts und Ablauf des Restart-Timers (Parameter RESTART-TIMER-SEC) nicht neu gestartet wurde.

Das Emergency-Kommando wird immer auf dem Rechner der überwachenden Knoten-Anwendung aufgerufen.

Der hier übergebene String wird von KDCDEF nicht weiter analysiert.

command_string1 darf bis zu 200 Zeichen lang sein. Die Angabe des Emergency-Kommandos ist Betriebssystem-spezifisch.

Bei der Installation von openUTM werden auf den Plattformen jeweils plattform-spezifische Muster mit dem Namen `UTM-C.EMERGENCY` bzw. `utm-c.emergency` ausgeliefert.

B *BS2000/OSD:*

B Der Name einer BS2000-SDF-Prozedur. Geben Sie die Prozedur an, wie
B beim SDF-Kommando ENTER-PROCEDURE beschrieben (siehe Hand-
B buch BS2000/OSD-BC Kommandos). Die aufgerufene SDF-Prozedur muss
B 4 Stellungsparameter besitzen. Für jeden der 4 Prozeduroperanden ist `%s`
B als Stellvertreter anzugeben.

B Beispiel:
B
B
```
EMERGENCY-CMD = 'FROM-FILE=*LIB(SYSLIB.EXAMPLE,
UTM-C.EMERGENCY),PROC-PAR=(%s,%s,%s,%s)'
```

B Sie können noch weitere Operanden vom ENTER-PROCEDURE-Kom-
B mando angeben.

B Beispiel:
B
B
```
EMERGENCY-CMD = 'FROM-FILE=*LIB(SYSLIB.EXAMPLE,EMERGENCY),
PROC-PAR=(%s,%s,%s,%s), LOGGING=*YES, JOBCLASS=<jobclass>
```

X *Unix-Systeme:*

X Der Name eines Unix Shell-Skripts. Sie müssen den Namen vollqualifiziert
X angeben.

X Beispiel:
X
```
EMERGENCY-CMD = '/opt/lib/utm60a00/32/shsc/utm-c.emergency'
```

W *Windows-Systeme:*

W Der Name eines Windows Command-Skripts. Sie müssen den Namen voll-
W qualifiziert angeben.

W
W

Beispiel:

```
EMERGENCY-CMD = 'c:\utm60¬shsc¬utm-c.emergency.cmd'
```

Aufruf der Prozedur bzw. des Skripts command_string1 im Anwendungslauf

Der Prozedur bzw. dem Skript *command_string1* werden vier Argumente übergeben, um damit den ausgefallenen Cluster-Knoten zu identifizieren und Maßnahmen zur Behebung einleiten zu können.

Die Argumente werden in folgender Reihenfolge übergeben:

APPLICATIONNAME	Name der UTM-Anwendung
FILEBASE	Filebase-Name der KDCFILE der ausgefallenen Knoten-Anwendung
HOSTNAME	Rechnername des ausgefallenen Knotens
VIRTUALHOST	Virtueller Rechnername des ausgefallenen Knotens

Der Returnwert der Prozedur bzw. des Skripts wird nicht bewertet.

B *BS2000/OSD:*

B
B Im BS2000/OSD werden die konfigurierten Prozeduren mit dem Kommando ENTER-PROCEDURE aufgerufen.

B Die aufgerufene BS2000-Prozedur muss 4 Stellungsparameter besitzen.

B
B
B Die Stellvertreterzeichen `%s` werden beim Aufruf durch die Werte von APPLICATIONNAME, FILEBASE, HOSTNAME und VIRTUALHOST in dieser Reihenfolge ersetzt.

X *Unix-Systeme:*

X
X Das generierte Skript *command_string1* wird als Hintergrundprozess gestartet.

X Das Skript *command_string1* wird mit folgenden Argumenten aufgerufen:

X APPLICATIONNAME FILEBASE HOSTNAME VIRTUALHOST

W *Windows-Systeme:*

W
W Das Kommando-Skript *command string* wird mit dem Windows-Kommando START aufgerufen ohne auf das Beenden zu warten.

W Es wird mit folgenden Argumenten aufgerufen:

W APPLICATIONNAME FILEBASE HOSTNAME VIRTUALHOST

FAILURE-CMD=command-string2

Mit diesem Operanden wird ein Kommando-String übergeben, der ein auszuführendes Kommando enthält.

command-string2 darf bis zu 200 Zeichen lang sein. Die Angabe des Failure-Kommandos ist Betriebssystem-spezifisch.

Das Failure-Kommando wird von openUTM aufgerufen, wenn eine Knoten-Anwendung abnormal beendet wurde, oder der Ausfall einer Knoten-Anwendung erkannt wurde. Über das Failure-Kommando kann ein Anwender z.B. einen Neu-Start der ausgefallenen Knoten-Anwendung veranlassen.

Das Failure-Kommando wird immer auf dem Rechner der überwachenden Knoten-Anwendung aufgerufen.

Ansonsten sind Syntax und Aufruf von FAILURE-CMD identisch zu Syntax und Aufruf von EMERGENCY-CMD (siehe Seite 302).

Bei der Installation von openUTM werden auf den Plattformen jeweils plattform-spezifische Muster mit dem Namen `UTM-C.FAILURE` bzw. `utm-c.failure` ausgeliefert.

FILE-LOCK-RETRY=number

Anzahl von Wiederholungen einer Sperranforderung für eine Cluster-globale Datei, falls die Sperre nicht in der im Parameter FILE-LOCK-TIMER-SEC vorgegebenen Zeit zugeteilt wurde.

Minimalwert: 0, d.h. keine Wiederholung
Maximalwert: 10
Standardwert: 0

FILE-LOCK-TIMER-SEC=time

Zeit in Sekunden, die eine Knoten-Anwendung einer UTM-Cluster-Anwendung maximal auf die Zuteilung einer Sperre für eine Cluster-globale Datei wartet.

Minimalwert: 1
Maximalwert: 60
Standardwert: 10

GLOBAL-UTM-DATA=

YES Die Verwendung der globalen UTM-Speicherbereiche GSSB und ULS ist in der UTM-Cluster-Anwendung erlaubt. Sie werden aber nur lokal unterstützt, d.h. in jeder Knoten-Anwendung existiert ein eigenes Exemplar des Speicherbereichs.

NO Die Verwendung der globalen UTM-Speicherbereiche GSSB und ULS ist in der UTM-Cluster-Anwendung verboten. Es dürfen keine GSSB oder ULS generiert werden. Versuche eines Teilprogramms, mit SPUT, SGET, SREL und UNLK auf diese UTM-Speicherbereiche zuzugreifen, führen zur abnormalen Vorgangsbeendigung mit KDCS-Returncode 70Z.

Standardwert: NO

RESTART-TIMER-SEC=time

Zeit in Sekunden, die eine Knoten-Anwendung nach einem Ausfall maximal für einen Warm-Start benötigt.

Nach einer Ausfallerkennung und dem Aufruf des Failure-Kommandos für eine ausgefallene Knoten-Anwendung zieht die überwachende Knoten-Anwendung einen Timer mit der hier angegebenen Zeit auf. Wenn die ausgefallene Knoten-Anwendung nach Ablauf dieser Zeit nicht wieder verfügbar ist, wird das Emergency-Kommando für die ausgefallene Knoten-Anwendung gestartet.

Bei Angabe des Wertes 0 wird der Neu-Start der ausgefallenen Knoten-Anwendung nicht zeitüberwacht.

Minimalwert: 0, d.h. keine Überwachung eines Anwendungs-Restarts
Maximalwert: 3600
Standardwert: 0

USER-RESTART=

YES USER mit RESTART=YES dürfen generiert werden. Der Vorgangskontext eines USERs ist nur in der Knoten-Anwendung verfügbar, in der er erzeugt wurde. Ein Vorgangswiederanlauf findet somit immer mit dem Vorgangskontext statt, der in dieser Knoten-Anwendung erzeugt wurde, d.h. in jeder Knoten-Anwendung kann ein eigenes Exemplar des Vorgangskontextes existieren. Bitte beachten Sie dabei, dass LSSBs zum Vorgangskontext gehören.

NO Es dürfen keine USER mit RESTART=YES generiert werden.

Standardwert: NO

B IMPORT-USER-LOCALES=

B B YES Die Angaben zu User Locales aus den Generierungsanweisungen für Benutzer werden von KDCDEF in die Cluster-User-Datei übernommen.

B B B NO Die Angaben zu User Locales aus den Generierungsanweisungen für Benutzer, die bereits in der Cluster-User-Datei enthalten sind, werden von KDCDEF nicht in die Cluster-User-Datei übernommen.

B Standardwert: NO

CLUSTER-NODE – Knoten-Anwendung einer UTM-Cluster-Anwendung definieren

Mit der CLUSTER-NODE-Anweisung konfigurieren Sie eine Knoten-Anwendung einer UTM-Cluster-Anwendung.

B B Ein XCS-Verbund in BS2000/OSD unterstützt maximal 16, gleichzeitig gestartete Knoten-Anwendungen.

X/W X/W In Unix-/Windows-Systemen können Sie für eine UTM-Cluster-Anwendung bis zu 32 Knoten-Anwendungen gleichzeitig starten.

Pro UTM-Cluster-Anwendung dürfen Sie die CLUSTER-NODE-Anweisung bis zu 32-mal angeben.

Wenn Sie eine UTM-Cluster-Anwendung generieren möchten, müssen Sie mindestens zwei CLUSTER-NODE-Anweisungen angeben. Wenn Sie eine CLUSTER-NODE-Anweisung angegeben haben, muss auch eine CLUSTER-Anweisung generiert werden.

Wenn Sie bei einer Neu-Generierung Angaben in der CLUSTER-Anweisung oder den CLUSTER-NODE-Anweisungen ändern, müssen Sie eine neue Cluster-Konfigurationsdatei erstellen (OPTION GEN=CLUSTER) und einsetzen, damit die Änderungen wirksam werden.

```
CLUSTER-NODE     FILEBASE = node_filebase
                ,HOSTNAME = host_name
               [ ,VIRTUAL-HOST = virtual_host_name]
```

B *zusätzlicher Operand in BS2000/OSD*

B `[ ,CATID = ( <catid_A> [ , <catid_B> ] ) ]`

FILEBASE = node_filebase

Basisname der KDCFILE, der Benutzer-Protokolldatei und der System-Protokolldatei SYSLOG dieser Knoten-Anwendung. Beim Start einer Knoten-Anwendung werden die UTM-Systemdateien unter dem hier angegebenen Namen erwartet. Die KDCFILE muss von allen Knoten-Anwendungen aus zugreifbar sein.

Dieser Operand ersetzt den Startparameter FILEBASE einer stand-alone UTM-Anwendung.

Die Basisnamen der einzelnen CLUSTER-NODE-Anweisungen müssen sich voneinander unterscheiden. Es gelten die gleichen Beschränkungen wie bei MAX KDCFILE=*filebase*.

Pflichtoperand.

B *BS2000/OSD:*

B Beim Start einer Knoten-Anwendung einer UTM-Cluster-Anwendung werden die UTM-Systemdateien dieser Knoten-Anwendung unter dem hier angegebenen Namen erwartet. Der *node_filebase* kann eine BS2000-UserId enthalten, aber er muss ohne CATID angegeben werden. CATIDs müssen Sie mit dem Operand CATID angeben. Der Name kann – unter Berücksichtigung von CATID und USERID – bis zu 42 Zeichen lang sein, siehe auch Abschnitt „BS2000/OSD:“ auf Seite 45.

X/W *Unix-/Windows-Systeme:*

X/W *node_filebase* bezeichnet das Dateiverzeichnis, das beim Start einer Knoten-Anwendung einer UTM-Cluster-Anwendung die KDCFILE und alle Dateien der Anwendung enthält. Der hier **angegebene Name** muss aus Sicht **aller Rechner des Clusters** das gleiche Dateiverzeichnis bezeichnen. Der Name kann bis zu 27 Zeichen lang sein.

HOSTNAME=host_name

Rechnername dieses Knotens. Geben Sie den primären Namen dieses Rechners an.

Der Name kann bis zu 64 Zeichen lang sein.

Die Rechnernamen der einzelnen CLUSTER-NODE-Anweisungen müssen sich voneinander unterscheiden. Rechnernamen, die sich nur durch Groß-/Kleinschreibung unterscheiden, werden als identisch betrachtet.

Pflichtoperand.

VIRTUAL-HOST=virtual_host_name

übernimmt für UTM-Cluster-Anwendungen die Funktion des Parameters MAX HOSTNAME. Den Parameter MAX HOSTNAME dürfen Sie in UTM-Cluster-Anwendungen nicht angeben.

B *BS2000/OSD:*

B Name des virtuellen Hosts, aus BCAM-Sicht, in dem die Knoten-Anwendung ablaufen soll.

B Standardwert: 8 Leerzeichen, d.h. die Knoten-Anwendung läuft unter dem Namen des realen Rechners.

X/W *Unix-/Windows-Systeme:*

X/W Durch die Angabe von HOSTNAME kann die Absenderadresse für Netzverbindungen spezifiziert werden, die von dieser Knoten-Anwendung aus aufgebaut werden. Standard sind 8 Leerzeichen. Dies bedeutet, dass bei Verbindungsaufbauten die Standard-Absenderadresse des Transportsystems

X/W
X/W
X/W

verwendet wird. Diese Funktion wird in einem Rechnerverbund benötigt, wenn beim Verbindungsaufbau als Absenderadresse die relocatable IP-Adresse und nicht die stationäre IP-Adresse verwendet werden soll.

B
B
B

CATID=(catid_A,catid_B)

Mit diesem Operanden legen Sie fest, welchen Katalogkennungen die KDCFILE-Dateien zugeordnet werden.

B
B
B
B

Dieser Operand ersetzt den Startparameter CATID einer stand-alone UTM-Anwendung. Beim Start einer Knoten-Anwendung einer UTM-Cluster-Anwendung werden die UTM-Systemdateien dieser Knoten-Anwendung in den hier angegebenen Katalogen erwartet.

B
B

Wenn die KDCFILE einfach geführt wird, müssen Sie nur *catid_A* angeben. *catid_B* ist in diesem Fall ohne Bedeutung.

B
B
B
B

Bei doppelter Dateiführung der KDCFILE wird Dateien mit dem Suffix *A* die CATID *catid_A* zugeordnet und Dateien mit dem Suffix *B* die CATID *catid_B*. Wenn Sie bei doppelter Dateiführung nur *catid_A* angeben, werden alle Dateien dieser CATID zugeordnet.

B

Standard: keine CATID

CON - Verbindung für die verteilte Verarbeitung über LU6.1 definieren

Die CON-Anweisung definiert eine Transportverbindung zwischen der lokalen UTM-Anwendung und einer Partner-Anwendung. Darüber hinaus ordnet sie der realen Partner-Anwendung einen LPAP-Partner zu, d.h. den logischen Anschlusspunkt der Partner-Anwendung in der lokalen Anwendung. Der LPAP-Partner muss mit einer LPAP-Anweisung definiert werden (siehe Seite 341).
Durch Angabe mehrerer CON-Anweisungen für ein und dieselbe Partner-Anwendung lassen sich parallele Transportverbindungen definieren.

Zur Generierung von LU6.1-Verbindungen siehe auch Abschnitt „Verteilte Verarbeitung über das LU6.1-Protokoll“ auf Seite 72.

Bei der Generierung der CON-, PTERM- und MUX-Anweisung ist zu beachten, dass das Namenstripel (*appliname* bzw. *ptermname*, *processorname*, *local_appliname*) innerhalb eines Generierungslaufs eindeutig sein muss.

Beispiel

Wurde bereits eine PTERM-Anweisung generiert mit

```
PTERM partnername1,PRONAM=processorname1,
```

so kann keine CON-Anweisung generiert werden mit

```
CON partnername1,PRONAM=processorname1,
```

wohl aber

```
CON partnername1,PRONAM=processorname1,BCAMAPPL=local_appliname1,
```

wenn *local_appliname1* nicht gleich dem primären UTM-Anwendungsnamen ist.

B Auch eine Anweisung `MUX` *partnername1* ... schließt eine Anweisung `CON` *partnername1* aus.

```
CON␣        remote_appliname
            [ ,BCAMAPPL=local_appliname ]
              ,LPAP=lpapname
              ,PRONAM={ processorname | C'processorname' }
                       nur in BS2000/OSD Pflichtoperand
            [ ,TERMN=termn_id ]

X/W         Unix- und Windows-System-spezifische Operanden
X/W         [ ,LISTENER-PORT=number ]
X/W         [ ,T-PROT=RFC1006 ]
X/W         [ ,TSEL-FORMAT={ T | E | A } ]
```

remote_appliname

Name der Partner-Anwendung, mit der über die logische Verbindung kommuniziert werden soll.

Für *remote_appliname* können Sie max. 8 abdruckbare Zeichen angeben. Zulässige Zeichen sind Groß- und Kleinbuchstaben, Ziffern und die Zeichen $, # und @. Bindestriche im Namen sind nicht erlaubt. Werden in dem Namen Kleinbuchstaben verwendet, dann muss der Name in Hochkommata ('...') eingeschlossen werden.

Die Angabe von *remote_appliname* ist Pflicht.

B *BS2000/OSD:*
B *remote_appliname* kann entweder der BCAM-Name einer UTM-Partner-Anwendung (bei homogener Kopplung) oder der Name einer TRANSIT-Anwendung (bei heterogener Kopplung) sein.
B Das erste Zeichen muss ein Großbuchstabe sein.

X/W *Unix-, Windows-Systeme*
X/W Bei *remote_appliname* müssen Sie den T-Selektor angeben, mit dem sich die
X/W Partner-Anwendung beim Transportsystem anmeldet.
X/W Das erste Zeichen muss ein Buchstabe sein.

BCAMAPPL=local_appliname

bezeichnet einen Namen der lokalen Anwendung, wie er in der Steueranweisung MAX oder BCAMAPPL festgelegt wurde. Es darf kein BCAMAPPL-Name angegeben werden, für den T-PROT=SOCKET generiert ist.

X/W Unter Unix- und Windows-Systemen darf der Name nicht mit '$' beginnen.

Der in der CLUSTER-Anweisung angegebene BCAMAPPL-Name ist hier nicht erlaubt.

Standard:
Wird keine Angabe gemacht, so gilt der primäre Anwendungsname in MAX ...,APPLINAME=.

X/W LISTENER-PORT=number
X/W Portnummer der Partner-Anwendung, wenn die Verbindungen zur Partner-
X/W Anwendung über TCP/IP aufgebaut wird.

X/W Erlaubte Werte: 102 und 1025 - 32767

X/W Standard: 0 (keine Portnummer)
X/W Bei OPTION CHECK-RFC1006=YES muss bei LISTENER-PORT eine
X/W Portnummer angegeben werden.

LPAP=lpapname

Name des LPAP-Partners der Partner-Anwendung, zu der die Verbindung aufgebaut werden soll. Der Name des LPAP-Partners, über den sich die Partner-Anwendung anschließt, muss mit der Anweisung LPAP *lpapname* definiert werden.

Durch Angabe mehrerer CON-Anweisungen mit gleichem *lpapname* werden parallele Verbindungen zur Partner-Anwendung aufgebaut.

Dabei müssen Sie darauf achten, dass die parallelen Verbindungen zu derselben Partner-Anwendung (*remote_appliname* und *processorname*) führen.

Pflichtparameter

PRONAM={ processorname | C'processorname' }

Name des Partner-Rechners. Der Name darf maximal 8 Zeichen lang sein. Enthält *processorname* Kleinbuchstaben oder Sonderzeichen, dann muss er als Zeichenkette mit C'...' angegeben werden.

B Pflichtoperand für BS2000/OSD

B *BS2000/OSD:*
B Für *processorname* ist der Name des Prozessors anzugeben, auf dem die
B Partner-Anwendung *remote_appliname* abläuft. Das ist entweder der Name
B eines Unix-Systems, eines Windows-Systems, eines BS2000-Rechners
B oder der TRANSDATA-Name eines Kommunikations- bzw. Vorrechners bei
B heterogener Kopplung. Der Name wird bei der Generierung des Netzes
B festgelegt. Fragen Sie den Netzadministrator.

B Sie müssen PRONAM= nicht angeben, wenn Sie zuvor mit der DEFAULT-
B Anweisung einen Standardwert dafür festgelegt haben.

X/W *Unix-, Windows-Systeme*:
X/W Für die Angabe von *processorname* haben Sie zwei Möglichkeiten:

X/W – Sie geben den realen Hostnamen an, unter dem die IP-Adresse des
X/W Partner-Rechners im Name Service des lokalen Systems (z.B. der
X/W hosts-Datei) eingetragen ist. Einen Aliasnamen des Rechners dürfen
X/W Sie nicht angeben.

X/W – Sie geben den UTM-Hostnamen des Partner-Rechners an.
X/W Voraussetzung hierfür ist, dass Sie die Umgebungsvariable
X/W UTM_NET_HOSTNAME gesetzt und den UTM-Hostnamen in der Um-
X/W wandlungsdatei festgelegt haben (siehe Abschnitt „Mapped Hostnamen verwenden (Unix- und Windows-Systeme)“ auf Seite 118).

X/W Standardwert: 8 Leerzeichen.

TERMN=termn_id

Maximal 2 Zeichen langes Kennzeichen für die Art des Kommunikationspartners. *termn_id* wird nicht von openUTM abgefragt, es wird vom Benutzer zur Auswertung gesetzt, um beispielsweise Terminaltypen abzufragen oder zu gruppieren etc. Das Kennzeichen *termn_id* wird im KB-Kopf für Auftragnehmer-Vorgänge eingetragen, d.h. für Vorgänge, die von einer Partner-Anwendung in der lokalen Anwendung gestartet wurden.

Standard: A4

X/W T-PROT= Adressformat, mit dem sich die Partner-Anwendung beim Transportsystem
X/W anmeldet. Informationen zu den folgenden Adressformaten finden Sie im
X/W CMX-Benutzerhandbuch oder in der Online-Hilfe zu PCMX.

X/W RFC1006 Adressformat RFC1006

X/W Standard: RFC1006

X/W TSEL-FORMAT=

X/W Formatindikator des T-Selektors. Der Formatindikator gibt die Codierung
X/W des T-Selektors im Transportprotokoll an. Nähere Informationen finden Sie
X/W im CMX-Benutzerhandbuch bzw. in der Online-Hilfe zu PCMX.

X/W T TRANSDATA-Format

X/W E EBCDIC-Zeichenformat

X/W A ASCII-Zeichenformat

X/W Standard:
X/W T wenn der Zeichenvorrat von *remote_appliname* dem TRANSDATA-For-
X/W mat entspricht
X/W E sonst

X/W Für den Betrieb über RFC1006 wird jedoch empfohlen, explizit einen Wert
X/W für TSEL-FORMAT anzugeben.

X/W **Adresse der Partner-Anwendung einer UTM-Anwendung unter Unix-, Windows-**
X/W **Systemen**

X/W Für den Aufbau einer Verbindung zur Partner-Anwendung, muss die UTM-Anwendung die
X/W Adresse der Partner-Anwendung kennen. Sie können Sie mit Hilfe der folgenden Operan-
X/W den angeben:

X/W – *remote_appliname* (Adresse der Partner-Anwendung im Partner-Rechner)
X/W – PRONAM (realer oder UTM-Hostname des Partner-Rechners)
X/W – LISTENER-PORT (Portnummer für RFC1006)
X/W – T-PROT (das verwendete Transportprotokoll)
X/W – TSEL-FORMAT (Formatindikator des T-Selektors)

X/W Siehe dazu Abschnitt „Adressinformationen für das Transportsystem CMX bereitstellen
X/W (Unix- und Windows-Systeme)“ auf Seite 105.

CREATE-CONTROL-STATEMENTS - KDCDEF-Steueranweisungen erzeugen

Der inverse KDCDEF ermöglicht es Ihnen, UTM-Objekte, die Sie im Anwendungsbetrieb dynamisch in die Konfiguration eingetragen haben, bei der Neugenerierung Ihrer Anwendung endgültig in die Konfiguration zu übernehmen. Siehe hierzu auch den Abschnitt „Inverser KDCDEF“ auf Seite 265.

Die Anweisung CREATE-CONTROL-STATEMENTS als erste Anweisung des KDCDEF-Laufs erzeugt für die dynamisch eingetragenen UTM-Objekte KDCDEF-Steueranweisungen und schreibt diese in die Datei *control_statements_file* (inverser KDCDEF). Sie können die Datei *control_statements_file* in ein und demselben KDCDEF-Lauf wieder als Generierungsgrundlage verwenden, indem Sie die Datei mit der Anweisung OPTION ...,DATA=*control_statements_file* als Inputdatei zuweisen.

Wenn die Datei *control_statements_file* mit den erzeugten KDCDEF-Steueranweisungen zu einem späteren Zeitpunkt in einem KDCDEF-Lauf verarbeitet werden soll, müssen Sie die Steueranweisungen in der Datei mit der Anweisung END abschließen.

Beim inversen KDCDEF können Steueranweisungen für UTM-Objekte des Typs TAC, PROGRAM, PTERM, LTERM, USER, CON, LTAC, LSES und KSET erzeugt werden.

Falls bei einer Neugenerierung mit KDCDEF zu einer PTERM- oder TPOOL-Anweisung mit PTYPE=APPLI, SOCKET oder UPIC-R in einer zugeordneten LTERM-Definition kein Benutzer angegeben ist, dann generiert KDCDEF implizit einen Benutzer mit dem LTERM-Namen. Für solche implizit generierten Benutzer werden jedoch vom inversen KDCDEF **weder** USER-Anweisungen erzeugt **noch** wird der Benutzername in der LTERM-Anweisung beim Operanden USER= eingefügt.

UTM-Objekte, die per Administration mit KC_DELETE_OBJECT zum Löschen vorgemerkt wurden, werden beim inversen KDCDEF nicht mehr in die *control_statements_file* übernommen. Nach einem KDCDEF-Lauf mit der Inputdatei *control_statements_file* können die Namen der gelöschten UTM-Objekte wieder neu verwendet werden.

Für den inversen KDCDEF können Sie KDCDEF mit mindestens einer Anweisung CREATE-CONTROL-STATEMENTS auch ohne weitere KDCDEF-Steueranweisungen starten.

Bei einem Versionswechsel müssen die KDCDEF-Steueranweisungen zunächst in der Vorgängerversion erzeugt werden, bevor die Steueranweisungen in einer höheren Version von dem Generierungstool KDCDEF verarbeitet werden können.

```
CREATE-CONTROL-STATEMENTS␣ { *ALL | CON | DEVICE | KSET | LSES | LTAC |
                             PROGRAM | USER }
                            ,FROM-FILE=kdcfile
                            ,TO-FILE=control_statements_file
                            [ ,MODE={ CREATE |EXTEND } ]
```

*ALL Es werden KDCDEF-Steueranweisungen für folgende Objektypen erzeugt:
- CON
- KSET
- LSES
- LTAC
- LTERM
- PROGRAM
- PTERM
- TAC
- USER

KDCDEF-Steueranweisungstypen für andere Objekttypen können nicht erzeugt werden.

CON Es werden KDCDEF-Steueranweisungen für die Transportverbindungen zu entfernten LU6.1-Anwendungen erzeugt.

DEVICE Es werden KDCDEF-Steueranweisungen für LTERM-Partner, Clients und Drucker erzeugt, d.h. für die Objekttypen:
- PTERM
- LTERM

KSET Es werden KDCDEF-Steueranweisungen für Keysets erzeugt, d.h. für Objekte vom Typ KSET.

LSES Es werden KDCDEF-Steueranweisungen für die Vergabe von LU6.1-Sessionnamen erzeugt.

LTAC Es werden KDCDEF-Steueranweisungen für Transaktionscodes erzeugt, über die Service-Programme in Partner-Anwendungen gestartet werden. Das sind Objekte vom Typ LTAC.

PROGRAM Es werden KDCDEF-Steueranweisungen für Programme, VORGANG-Exits, Transaktionscodes und TAC-Queues erzeugt, d.h. für die Objekttypen:
- TAC
- PROGRAM

USER Es werden KDCDEF-Steueranweisungen für Benutzerkennungen erzeugt, d.h. Objekte vom Typ USER.

Für Benutzerkennungen ist zu beachten, dass der inverse KDCDEF Passwörter nicht rekonstruiert. Für Benutzer mit Passwörtern werden Anweisungen wie folgt erzeugt: `USER username, PASS=*RANDOM, ...`
Bei stand-alone Anwendungen müssen Sie nach Beendigung des KDCDEF-Laufs die Passwörter mit dem Tool KDCUPD in die neue KDCFILE übertragen. Dies ist auch bei der Generierungsvariante UTM-F möglich.

FROM-FILE=kdcfile
Name der KDCFILE, aus der die Steueranweisungen erzeugt werden.

Die openUTM-Versionen der KDCFILE und des Generierungstools KDCDEF müssen übereinstimmen.

TO-FILE=control_statements_file
Die erzeugten KDCDEF-Steueranweisungen werden in die Datei *control_statements_file* geschrieben. *control_statements_file* muss ein gültiger Dateiname sein. Die Datei *control_statements_file* wird mit der Anweisung OPTION ...,DATA=*control_statements_file* als Inputdatei für den KDCDEF-Lauf zugewiesen.

MODE= Schreibmodus der Datei, die die erzeugten KDCDEF-Steueranweisungen enthält.

CREATE
Die Datei *control_statements_file* wird erzeugt.

B
B
B
In BS2000/OSD erzeugt der inverse KDCDEF eine SAM-Datei. Falls bereits eine Datei gleichen Namens existiert, muss es eine SAM-Datei sein. Die bereits existierende SAM-Datei wird dann überschrieben.

X/W
X/W
Existiert in Unix- bzw. Windows-Systemen bereits eine Datei gleichen Namens, dann wird sie überschrieben.

EXTEND
Die erzeugten Steueranweisungen werden an die bereits existierende Datei *control_statements_file* angehängt. Falls die Datei noch nicht existiert, wird sie erzeugt.

DATABASE - Datenbanksystem definieren (BS2000/OSD)

Mit dieser Anweisung werden die Datenbanksysteme beschrieben, mit denen die UTM-Anwendung koordiniert zusammenarbeiten soll.

Für jedes Datenbanksystem muss eine DATABASE-Anweisung abgesetzt werden. Durch mehrere DATABASE-Anweisungen für ein Datenbanksystem können Sie einem Datenbanksystem mehrere Entry-Namen zuordnen.

Die Anweisung DATABASE darf mehrfach angegeben werden. Dadurch können maximal 2 (auf Sonderfreigabe maximal 8) unterschiedliche Datenbanksysteme definiert werden.

```
DATABASE      [ ENTRY=entryname ]
              [ ,LIB= { omlname | LOGICAL-ID(logical-id) } ]
              [ ,TYPE={ UDS | SESAM | PRISMA | LEASY | CIS | DB | XA} ]
```

ENTRY=entryname

Entry-Name der Datenbank. Es gelten die folgenden Standardwerte:

$UNIBASE	bei TYPE = UDS
SESAM	bei TYPE = SESAM
PRISCON	bei TYPE = PRISMA
LEASY	bei TYPE = LEASY
CIS	bei TYPE = CIS
DB	bei TYPE = DB

Bei einer Generierung des XA-Anschlusses mit TYPE=XA in openUTM unter BS2000/OSD **muss** mit dem Parameter ENTRY der Name des XA-Switches angegeben werden, wie er vom Datenbank-System zur Verfügung gestellt wird. Mit der DATABASE-Anweisung ist eine Generierung mehrerer XA-Switches möglich.

Ein Datenbankanschluss an Oracle muss mittels TYPE = XA generiert werden.

Weitere Entry-Namen (z. B. SQLUDS für UDS/SQL) sind in den Handbüchern zu den jeweiligen Datenbanksystemen zu finden.

B B LIB= Gibt die Bibliothek an, aus der das Verbindungsmodul zu dem Datenbanksystem dynamisch nachgeladen wird.

B B B omlname
Name der OML, aus der das Verbindungsmodul nachgeladen wird.
omlname kann max. 54 Zeichen lang sein.

B B B B B B LOGICAL-ID(logical-id)
Gibt an, dass das Verbindungsmodul im IMON-Installationspfad für das Datenbanksystem gesucht und von dort aus nachgeladen wird.
logical-id ist ein maximal 15 Zeichen langer Name. Er darf derzeit nur für SESAM/SQL und UDS/SQL angegeben werden; für beide DB-Systeme lautet er SYSLNK, siehe auch Hinweise auf Seite 318.

B B B B B Geben Sie LIB= nicht an, dann wird LIB= mit TASKLIB vorbelegt. Dies entspricht nicht dem Kommando SET-TASKLIB, sondern es muss dann eine Bibliothek mit dem Namen TASKLIB existieren. Ein Nachladen des Verbindungsmoduls aus der mit SYSFILE-TASKLIB zugewiesenen Bibliothek wird nicht unterstützt.

B B B B B B Beim Nachladen sucht der DBL das Verbindungsmodul zuerst in der Bibliothek, die Sie in LIB= angegeben haben. Existiert die Bibliothek nicht, dann bricht der DBL die Suche ab. Ist die Bibliothek zwar vorhanden, das Verbindungsmodul wird dort aber nicht gefunden, dann durchsucht der DBL die alternativen Bibliotheken. Das sind die Bibliotheken, denen ein Dateikettungsname BLSLIB*nn* (0≤*nn*≤99) zugewiesen wurde.

B B B B Werden mehrere DATABASE-Anweisungen mit demselben TYPE gegeben, um verschiedene Entries für eine Datenbank zu definieren, dann wird das Verbindungsmodul aus der Bibliothek geladen, die im Operanden LIB der ersten DATABASE-Anweisung des jeweiligen TYPE angegeben wurde.

B *Hinweise zur Verwendung von LOGICAL-ID*

B B B B B – LIB=LOGICAL-ID(*logical-id*) darf nur angegeben werden, wenn das Datenbanksystem ordnungsgemäß mit IMON installiert wurde. Wenn das Datenbanksystem nicht mit IMON installiert wurde, dann müssen Sie entweder das Verbindungsmodul statisch dazu binden (ohne Operanden LIB=) oder LIB=*omlname* angeben.

B B B – Die Angabe von LOGICAL-ID(*logical-id*) hat gegenüber *omlname* den Vorteil, dass die UTM-Anwendung unabhängig ist von Installationspfaden und Bibliotheksnamen des Datenbanksystems.

B B B – Wenn Sie LIB=LOGICAL-ID(SYSLNK) angeben und wenn mehrere Produktversionen installiert sind, dann wird standardmäßig die höchste Version genommen.

– Soll nicht die höchste Version nachgeladen werden, dann geben Sie entweder mit LIB=*omlname* die Bibliothek der niedrigeren Version an oder Sie weisen die Version vor dem Start der UTM-Anwendung zu (IMON-Kommando SELECT-PRODUCT-VERSION).

– Tritt beim Suchen des Verbindungsmoduls im IMON-Installationspfad ein Fehler auf, dann wird der Start der Anwendung abgebrochen und der Fehler nach SYSOUT protokolliert.

TYPE= Kennzeichnet das Datenbanksystem.

Mit TYPE=DB können Sie auch andere als die oben genannten Datenbank-Systeme anschließen, Voraussetzung ist, dass diese Datenbank-Systeme die Schnittstelle IUTMDB unterstützen.

Standard: UDS

DEFAULT - Standardwerte definieren (BS2000/OSD)

B Mit einer DEFAULT-Anweisung können Sie eigene Standardwerte für Operanden einer KDCDEF-Steueranweisung festlegen. Ein mit DEFAULT eingestellter Standardwert für einen Operanden gilt bis zur nächsten DEFAULT-Anweisung für die gleiche Steueranweisung und den gleichen Operanden. Um einen mit DEFAULT definierten Standardwert auf die UTM-Standardeinstellung zurückzusetzen, muss jeweils der UTM-Standardwert mit einer DEFAULT-Anweisung zugewiesen werden. Ist dies nicht explizit möglich, beispielsweise bei FORMAT= Leerzeichen, dann wird der Standardwert durch die Angabe von (STD) oder *STD eingestellt.

B Vorteil dieser Anweisungs-spezifischen Standardwerte: Wenn Sie eine Steueranweisung mehrmals angeben (z.B. PTERM), so brauchen Sie nicht in jeder dieser Anweisungen die gleichen Operandenwerte anzugeben (z.B. in PRONAM den Rechnernamen).

B Bei der Portierung von UTM-Anwendungen von BS2000/OSD auf Unix- bzw. Windows-Systemen ist zu beachten, dass die DEFAULT-Anweisung von openUTM unter Unix- und Windows-Systemen nicht unterstützt wird.

B
```
DEFAULT␣      control-statement␣operand [ ,operand ] [ ,... ]
```

B control-statement

ist die KDCDEF-Steueranweisung, für die mit dieser DEFAULT-Anweisung neue Standardwerte festgelegt werden sollen. Die folgenden Operanden sind abhängig von dieser Steueranweisung und beziehen sich nur auf die durch sie bezeichnete Klasse von Steueranweisungen. Dabei ist zu beachten, dass die PROGRAM- und AREA-Anweisungen **eine** Klasse bilden, d.h. geänderte Standardwerte der PROGRAM-Anweisung beziehen sich auch auf die anderen Anweisungen dieser Klasse.

Zwischen dem Operanden *control-statement* und den folgenden Operanden muss mindestens ein Leerzeichen stehen. Welche Steueranweisungen Sie hier angeben dürfen, entnehmen Sie der Tabelle auf der nächsten Seite.

operand ,... gibt einen (oder mehrere) Operanden der KDCDEF-Steueranweisung *control-statement* an. Die Operanden werden durch Kommata getrennt angegeben. Welche Operanden für *control-statement* jeweils zulässig sind, können Sie der Tabelle auf der nächsten Seite entnehmen.

	zulässige Steueranweisungen	zulässige Operanden
B B B B	CON	BCAMAPPL={*local_appliname* | (STD)} LPAP={*lpapname* | (STD)} PRONAM={*processorname* | C'*processorname*'} TERMN=*termn_id*
B B B B	LPAP	NETPRIO=*netprio* QLEV=*queue_level_number* STATUS={ON | OFF} SESCHA=*sescha_name*
B	LSES	LPAP=*sessionname*
B B B B B	LTAC	LPAP=*lpapname* LTACUNIT=*ltacunit* STATUS={ON | OFF} TYPE={D|A} WAITTIME=(*time1*,*time2*)
B B B B B B B B B B B B B	LTERM	ANNOAMSG={Y|N} FORMAT={*formatname* | (STD)} KERBEROS-DIALOG={YES | NO} LOCALE={ ([*lang_id*], [*terr_id*],[*ccsname*]) | *STD} NETPRIO=*netprio* PLEV=*print_level_number* QAMSG={Y|N|(STD)} QLEV=*queue_level_number* RESTART={YES | NO} STATUS={ON | OFF} USAGE={D | O}
B B B	LOAD-MODULE	LIB=*libname* LOAD-MODE=*loadmode* VERSION=*version*
B B	OSI-CON	ACTIVE={YES | NO} LOCAL-ACCESS-POINT=*access_point_name*
B B B B B	OSI-LPAP	APPLICATION-CONTEXT=*application_context* IDLETIME=*time* QLEV=*queue_level_number* STATUS={ON | OFF} TERMN=*termn_id*
B B	PROGRAM	COMP=*compiler* LOAD-MODULE={*lmodname* | *STD}

	zulässige Steueranweisungen	zulässige Operanden
B	PTERM	BCAMAPPL=*local_appliname* CONNECT={A | N} ENCRYPTION-LEVEL={NONE | 1 | 2 | 3 | 4 | TRUSTED} IDLETIME=*time* MAP={USER | SYSTEM | SYS | SYS1 | SYS2 | SYS3 | SYS4} PRONAM={*processorname* | C'*processorname*' | *RSO} PROTOCOL={N | STATION} PTYPE={*partnertyp* | *RSO | *ANY} STATUS={ON | OFF} TERMN={*termn_id* | (STD)} USAGE={D | O} USP-HDR={ALL | MSG | NO}
B	SESCHA	CONNECT={Y | N} CONTWIN={Y | N | (STD)} DPN={*instance_name* | (STD)} IDLETIME=*time* PLU={Y|N | (STD)} PACCNT=*number*
B	TAC	ADMIN={Y | N} CALL={BOTH | FIRST | NEXT | (STD)} DEAD-LETTER-Q={NO | YES} ENCRYPTION-LEVEL={NONE | 1 | 2} EXIT={*exit* | (STD) } PGWT={NO | YES} PROGRAM={*programname* | (STD)} QLEV=*queue_level_number* QMODE = {STD | WRAP-AROUND} RUNPRIO=*priority* SATADM={NO | YES} SATSEL={BOTH | SUCC | FAIL | NONE} STATUS={ON | OFF | HALT | KEEP} TACCLASS={*class* | (STD)} TACUNIT=*tacunit* TCBENTRY={*name_of_tcbentry-statement* | (STD)} TIME={*time1* | (*time1*,*time2*)} TYPE={D | A | Q}

	zulässige Steueranweisungen	zulässige Operanden
B	TPOOL	ANNOAMSG={ Y | N }
B		BCAMAPPL=*appliname*
B		ENCRYPTION-LEVEL={ NONE | 1 | 2 | 3 | 4 | TRUSTED}
B		FORMAT={*formatname* | (STD)}
B		IDLETIME=*time*
B		KERBEROS-DIALOG={YES | NO}
B		LOCALE={ ([*lang_id*], [*terr_id*],[*ccsname*]) | *STD }
B		MAP={ USER | SYSTEM | SYS | SYS1 | SYS2 | SYS3 | SYS4 }
B		NETPRIO={ MEDIUM | LOW }
B		NUMBER=*number1*
B		PRONAM={ *processorname* | C'*processorname*' | *ANY }
B		PROTOCOL={ N | STATION }
B		PTYPE={ *partnertyp* | *ANY }
B		QLEV=*queue_level_number*
B		TERMN={ *termn_id* | (STD) }
B		USP-HDR={ALL | MSG | NO}
B	USER	FORMAT={*formatname* | (STD)}
B		LOCALE={ ([*lang_id*], [*terr_id*],[*ccsname*]) | *STD }
B		PERMIT={NONE | ADMIN | SATADM | (ADMIN,SATADM)}
B		PROTECT-PW=(*length,level_of_complexity,max_time,min_time*)
B		QLEV=*queue_level_number*
B		QMODE = {STD | WRAP-AROUND}
B		RESTART={YES | NO}
B		SATSEL={BOTH | SUCC | FAIL | NONE}
B		STATUS={ON | OFF}

EDIT - Editoptionen definieren (BS2000/OSD)

Mit der EDIT-Anweisung lassen sich Bildschirmfunktionen und Eigenschaften der Bildschirmausgabe im Zeilenmodus (Editoptionen) zu Gruppen - so genannten Editprofilen - zusammenfassen und mit einem Namen versehen. Über diesen Namen kann ein Satz von Editoptionen aus einem Teilprogramm heraus angesprochen werden.

Die EDIT-Anweisung kann innerhalb eines Generierungslaufs mehrfach angegeben werden. In jeder EDIT-Anweisung muss jedoch ein anderer Name angegeben werden (Operand *name*).

Die Namen der Editprofile werden an der Programmierschnittstelle mit den Aufrufen MPUT, MGET, DPUT, FPUT und FGET im Feld KCMF angegeben, wobei als Formatsteuerzeichen ein Leerzeichen eingetragen wird.

openUTM interpretiert die Angaben im Feld KCMF wie folgt:

keine Editprofile generiert	Editprofile generiert
Ist als Formatsteuerzeichen ein Leerzeichen eingetragen, dann ignoriert openUTM die übrigen Zeichen des Feldes	Ist als Formatsteuerzeichen ein Leerzeichen eingetragen, dann müssen die übrigen Zeichen des Feldes KCMF entweder den Namen eines gültigen Editprofils oder aber Leerzeichen enthalten

Eine ausführlichere Beschreibung der im Folgenden beschriebenen Operanden findet sich im Benutzerhandbuch TRANSDATA TIAM. Nähere Informationen zum Arbeiten mit Editprofilen finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“.

```
EDIT␣          name
               [ ,BELL={ NO | YES } ]
               [ ,CCSNAME=ccsname ]
               [ ,HCOPY={ NO | YES } ]
               [ ,HOM={ NO | YES } ]
               [ ,IHDR={ NO | YES } ]
               [ ,LOCIN={ NO | YES } ]
               [ ,LOW={ YES | NO} ]
                 ,MODE={ EXTEND | LINE | PHYS | TRANS }
               [ ,NOLOG={ NO | YES } ]
               [ ,OHDR={ NO | YES } ]
               [ ,SAML={ NO | YES } ]
               [ ,SPECIN={ C | I | N } ]
```

name Hier muss ein max. 7 Zeichen langer alphanumerischer Name für den zu definierenden Satz von Editoptionen angegeben werden.

BELL= gibt an, ob bei der Ausgabe der Nachricht ein akustischer Alarm am Terminal ausgelöst werden soll.

CCSNAME=ccsname
(**c**oded **c**haracter_**s**et_**name**)
Für *ccsname* ist der max. 8 Zeichen lange Name des Zeichensatzes (CCS-Name) anzugeben, der für die Aufbereitung einer Nachricht verwendet wird. Der angegebene CCS-Name muss zu einem im BS2000-System definierten EBCDIC-Zeichensatz gehören (siehe auch Benutzerhandbuch zu XHCS). Der Zeichensatz muss kompatibel sein zu einem ISO-Zeichensatz, der von dem Terminal, an das die Nachricht gerichtet ist, unterstützt wird. Zum Zeitpunkt der Generierung kann KDCDEF weder die Gültigkeit des CCS-Namens im BS2000-System noch die Kompatibilitätsbedingung überprüfen.

Dem Editprofil darf kein CCS-Name zugeordnet werden, wenn Sie für den Operanden MODE den Wert TRANS (transparent Mode) angeben.

Wird das Editprofil zur Ausgabe von Nachrichten an einen RSO-Drucker verwendet, wird nur der Parameter CCSNAME= des Editprofils ausgewertet.

HCOPY= (**h**ard_**copy**)
gibt an, ob die Ausgabe-Nachricht zusätzlich zur Ausgabe auf dem Terminal auf einem dort angeschlossenen Hardcopy-Drucker protokolliert werden soll.

HOM= (**hom**ogeneous)
gibt an, ob die Ausgabe-Nachricht unstrukturiert, d.h. homogen, ausgegeben werden soll. Ist NO angegeben, wird die Nachricht strukturiert, d.h. nicht homogen ausgegeben. In diesem Fall wird eine logische Zeile als Ausgabeeinheit betrachtet.

IHDR= (**i**nput **h**ea**d**e**r**)
gibt an, ob der Nachrichtenvorspann der Eingabe-Nachricht an das Teilprogramm übergeben werden soll.

LOCIN= (**loc**al parameter **in**put)
Nur an Terminals, die lokale Parameter unterstützen (z.B. 9763), ist dieser Operand von Bedeutung. Bei der Angabe YES werden lokale Attribute in der Eingabe-Nachricht als logische Steuerzeichen an den Anwender weitergereicht. Bei NO werden lokale Attribute aus der Eingabe-Nachricht entfernt und nicht weitergegeben. LOCIN=YES ist nur in Verbindung mit MODE=EXTEND zulässig.

LOW= (**low**er case)
gibt an, ob die Eingabe-Nachricht, die an das Teilprogramm übergeben wird, auch Kleinbuchstaben enthalten kann. Ist NO angegeben, werden die Kleinbuchstaben vom System in Großbuchstaben umgesetzt.

MODE=

EXTEND (**extend**ed linemode)
gibt an, ob die Nachricht im „extended linemode“ ausgegeben werden soll. Ist MODE=EXTEND angegeben, darf nur bei den Editoptionen BELL, LOW und LOCIN der Wert YES angegeben werden. Für den Operanden SPECIN ist nur die Angabe N zulässig.

LINE (**line** mode)
Die Nachricht soll im Zeilenmodus ausgegeben werden. Sie kann durch logische Steuerzeichen strukturiert werden. Die Nachricht wird vom System aufbereitet. Ist MODE=LINE angegeben, ist für IHDR, OHDR und LOCIN nur die Angabe NO zulässig.

PHYS (**phys**ical mode)
Die Nachricht soll physikalisch, d.h. ohne Aufbereitung durch das System, ausgegeben bzw. eingelesen werden. Ist MODE=PHYS angegeben, darf nur bei den Editoptionen IDHR, LOW und OHDR der Wert YES angegeben werden. Für den Operanden SPECIN ist nur die Angabe N zulässig.

Für Nachrichten an Drucker ist diese Angabe nicht sinnvoll. Physikalische Nachrichten an Drucker können nur über Format-Exit realisiert werden.

TRANS (**trans**parent mode)
Die Ausgabe-Nachricht soll transparent übertragen werden. Ist MODE=TRANS angegeben, darf für keine weitere Editoption der Wert YES angegeben werden.

Für den Operanden SPECIN= ist nur die Angabe N zulässig.
Der Operand CCSNAME= darf nicht angegeben werden.

NOLOG= (**no log**ical characters)
gibt an, wie nicht abdruckbare Zeichen vom System behandelt werden sollen.

YES Die logischen Steuerzeichen werden nicht ausgewertet. Alle Zeichen, die im EBCDIC-Code kleiner X’40’ sind, werden durch Ersatzzeichen (SUB) ersetzt. Nur abdruckbare Zeichen werden durchgelassen.

NO Alle logischen Steuerzeichen werden ausgewertet. Spezielle physikalische Steuerzeichen werden durchgelassen. Andere Zeichen kleiner X’40’ werden durch Ersatzzeichen (SUB) ersetzt. Abdruckbare Zeichen werden durchgelassen.
Standard: NO

OHDR= (**o**utput **h**ea**d**e**r**)
gibt an, ob die Ausgabe-Nachricht einen Nachrichtenkopf enthält. Die Länge des Nachrichtenkopfs +1 muss im ersten Byte der Nachricht binär angegeben werden.

SAML= (**sam**e **l**ine)
ist nur für Drucker von Bedeutung.
Bei SAML=YES wird am Nachrichtenanfang kein Zeilenvorschub ausgeführt. Bei SAML=NO beginnt die Nachricht am Anfang der nächsten Zeile.

SPECIN= (**spec**ial **in**put)

C (**c**onfidential)
gibt an, ob die Eingabedaten am Terminal dunkel gesteuert dargestellt werden und damit unsichtbar bleiben.

I (**i**d-card)
gibt an, ob die nächste Eingabe über den Ausweisleser erfolgen soll.

N (**n**ormal)
Es wird eine normale Eingabe vom Terminal verlangt.

EJECT - Seitenvorschub im Protokoll veranlassen

Die Steueranweisung bewirkt einen Seitenvorschub im Protokoll. Die EJECT-Zeile wird nicht mitprotokolliert und nicht mitgezählt.

```
EJECT␣
```

END - KDCDEF-Eingabe beenden

Diese Steueranweisung kennzeichnet das Ende der Steueranweisungen und muss als letzte Anweisung eingegeben werden.

```
END␣
```

Enthält eine Datei, die mit OPTION DATA=*filename* als KDCDEF-Inputdatei angegeben wurde, eine END-Anweisung, so wird die KDCDEF-Eingabe nach Bearbeitung dieser Anweisung unmittelbar beendet.

EXIT - Event-Exits definieren

Mit der EXIT-Anweisung werden die Event-Exits definiert, die in der Anwendung verwendet werden.

Für den Event-Exit FORMAT dürfen Sie pro KDCDEF-Lauf nur eine EXIT-Anweisung angeben. Für den Event-Exit INPUT dürfen Sie pro Typ nur eine EXIT-Anweisung angeben. Für die Event-Exits START und SHUT dürfen Sie jeweils bis zu acht EXIT-Anweisungen angeben. Diese EXIT-Anweisungen müssen sich jedoch bezüglich der Angabe für den Operanden PROGRAM= unterscheiden.

Beim Start bzw. beim Ende eines UTM-Prozesses werden alle als START- bzw. SHUT-Exit definierten Programme nacheinander aufgerufen. Dabei bestimmt die Reihenfolge der EXIT-Anweisungen im KDCDEF-Lauf die Reihenfolge, in der openUTM die START- bzw. SHUT-Exit-Programme aktiviert .

Nähere Informationen zu Event-Exits finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“.

B *Event-Exits in BS2000/OSD:*
B Die Event-Exits START, SHUT, INPUT und FORMAT dürfen keinem Lademodul zugeordnet
B werden, das mit LOAD-MODULE LOAD-MODE=ONCALL generiert ist.

Die Event Services MSGTAC, BADTACS und SIGNON müssen mit TAC-Anweisungen definiert werden.

```
EXIT␣        PROGRAM=objectname

             ,USAGE={ START |
                      SHUT |
                      ( INPUT, { ALL | FORMMODE | LINEMODE | USERFORM[1] } ) |
                      FORMAT[1] }
```

B [1] FORMAT und USERFORM sind nur in BS2000/OSD erlaubt.

PROGRAM=name

Name des Programms, das die für diesen Event-Exit auszuführenden Funktionen enthält. Es muss eine PROGRAM-Anweisung mit diesem Namen (*objectname*) angegeben werden.

USAGE= Typ des Event-Exits

START wird als Event-Exit START eingesetzt.

SHUT wird als Event-Exit SHUT eingesetzt.

INPUT wird als Event-Exit INPUT eingesetzt.
Zusätzlich müssen Sie den Typ des INPUT-Exits angeben:

ALL Event-Exit INPUT, der Nachrichten aller Formatsteuerzeichen sowie LINEMODE-Nachrichten behandelt.

Wird als Typ ALL angegeben, dann handelt es sich um den einzigen Event-Exit INPUT der Anwendung. Es dürfen damit keine weiteren Event-Exits INPUT generiert werden.

FORMMODE
Event-Exit INPUT für +, * und #Formate

LINEMODE
Event-Exit INPUT für LINEMODE-Nachrichten

B B USERFORM
Event-Exit INPUT für -Formate

B FORMAT wird als Event-Exit FORMAT eingesetzt

FORMSYS - Formatierungssystem beschreiben

Die Steueranweisung FORMSYS beschreibt das Formatierungssystem.
Nur die erste FORMSYS-Anweisung wird ausgewertet.

X/W X/W In Anwendungen auf Unix- und Windows-Systemen kann die FORMSYS-Anweisung entfallen, da nur der Standardwert NONE angegeben werden kann.

```
FORMSYS␣      [ TYPE=typ ]

              zusätzliche Operanden in BS2000/OSD
              [ ,ENTRY=entryname ]
              [ ,LIB=omlname ]
```

TYPE=typ Angabe für das Formatierungssystem:

– In BS2000/OSD kann derzeit nur FHS angegeben werden.

Standard (BS2000/OSD): FHS

– In Unix- und Windows-Systemen kann nur NONE angegeben werden. NONE bedeutet, dass ohne Formatierungsfunktion gearbeitet wird.

Standard (Unix- und Windows-Systeme): NONE

ENTRY=entryname
Entry-Name für das Formatierungssystem

Standardwert: KDCFHS bei TYPE=FHS

LIB=omlname bezeichnet die Objektmodulbibliothek (OML), aus der das Anschlussmodul zum Formatierungssystem geladen wird. *omlname* kann max. 54 Zeichen lang sein.

Geben Sie LIB= nicht an, dann wird LIB= mit TASKLIB vorbelegt. Dies entspricht nicht dem Kommando SET-TASKLIB, sondern es muss dann eine Bibliothek mit dem Namen TASKLIB existieren. Ein Nachladen des Anschlussmoduls aus der mit SYSFILE-TASKLIB zugewiesenen Bibliothek wird nicht unterstützt.

i Beim Nachladen sucht der DBL das Anschlussmodul zuerst in der Bibliothek, die Sie in LIB= zugewiesen haben. Existiert die Bibliothek nicht, dann bricht der DBL die Suche ab. Der DBL bricht die Suche nicht ab, wenn LIB nicht angegeben wurde, sondern mit TASKLIB vorbelegt wurde und keine Datei mit diesem Namen existiert.
Ist die Bibliothek zwar vorhanden, das Anschlussmodul wird dort aber nicht

B
B
B

gefunden, dann durchsucht der DBL die alternativen Bibliotheken. Das sind die Bibliotheken, denen ein Dateikettungsname BLSLIB*nn* (0≤*nn*≤99) zugewiesen wurde.

KSET - Keyset definieren

Mit der Steueranweisung KSET werden die Key- oder Zugangscodes einer Anwendung, die für den Zugriffsschutz definiert wurden, zu einem logischen Keyset zusammengefasst. Für ein Keyset dürfen mehrere Steueranweisungen angegeben werden.

KDCDEF erzeugt implizit das Keyset KDCAPLKS, das sämtliche Keycodes enthält.

```
KSET␣        keysetname
              ,KEYS={ ( key1,key2,... key n) | MASTER }
```

keysetname Name des definierten Keysets.
keysetname kann max. 8 Zeichen lang sein.

Das Keyset kann zugeordnet werden:
- einem Benutzer (in der USER-Anweisung auf Seite 542)
- einem LTERM-Partner (in der LTERM-Anweisung auf Seite 358)
- einer Partner-Anwendung (in der LPAP- bzw. OSI-LPAP-Anweisung auf Seite 342 bzw. Seite 436)
- einem TAC (in der TAC-Anweisung auf Seite 498)
- einem LTAC (in der LTAC-Anweisung auf Seite 348)
- einem TPOOL (in der TPOOL-Anweisung auf Seite 526 bzw. Seite 533)

Dem Client oder der Partner-Anwendung steht nach dem Aufbau der Verbindung der Keyset des LTERM- oder (OSI-)LPAP-Partner, der der Verbindung zugeordnet ist, zur Verfügung. Nach der Anmeldung an die Anwendung steht dem Client oder der Partner-Anwendung der Keyset der Benutzerkennung zur Verfügung.
Das Lock-/Keycode- und das Access-List-Konzept werden ausführlich im openUTM-Handbuch „Konzepte und Funktionen“ beschrieben. Eine Einführung in die Zugriffskontrolle finden Sie im gleichnamigen Abschnitt „Lock-/Keycode-Konzept“ auf Seite 216.

KEYS= Key- oder Zugangscodes des Keysets *keysetname*

(key1,..., key*n*)

Liste von Zahlen zwischen 1 und dem in der Anwendung erlaubten Maximalwert (MAX ...,KEYVALUE=*number*). Die Zahlen entsprechen den Keycodes, die in diesem Keyset zusammengefasst werden.

Ein Key- oder Zugangscode berechtigt zum Zugriff auf eine mit einem Lockcode bzw. einer Access-List gesicherten Ressource, wenn Keycode und Lockcode übereinstimmen bzw. der Zugangscode in der Access-List enthalten ist.

Pro KSET-Anweisung können maximal 60 Keycodes/Zugangscodes angegeben werden. Soll ein Keyset mehr als 60 Codes enthalten, wird eine weitere KSET-Anweisung mit demselben *keysetname* angegeben.

Die Klammer kann weglassen werden, wenn nur ein Code angegeben wird.

openUTM ignoriert den Wert 0 für *key* ohne Meldung.

MASTER Das MASTER-Keyset enthält alle Keycodes/Zugangscodes der Anwendung.

LOAD-MODULE - Lademodule (BLS) beschreiben (BS2000/OSD)

Mit der LOAD-MODULE-Anweisung werden Namen, Version und Eigenschaften von Lademodulen festgelegt. Mit LOAD-MODULE-Anweisungen sind alle Lademodule zu beschreiben, die während des Programmlaufs mit BLS ausgetauscht, bzw. als eigenständige Einheiten nachgeladen werden. Für jedes Lademodul ist eine eigene LOAD-MODULE-Anweisung zu schreiben.

Die Lademodule, die Sie mit dem BLS bearbeiten können, sind entweder LLMs (Link and Load Module) oder OMs (Object Module). Es wird jedoch empfohlen, die nachzuladenden Programmteile und Datenbereiche zu LLMs zu binden (siehe Handbücher zu BLS).

Ein Lademodul kann mehrere Teilprogramme und Datenbereiche enthalten. Die Teilprogramme und Datenbereiche können mit PROGRAM- bzw. AREA-Anweisungen beschrieben werden. Sie können einer LOAD-MODULE-Anweisung eine oder mehrere PROGRAM- und/oder AREA-Anweisungen zuordnen. Die Zuordnung erfolgt über den Namen des Lademoduls *lmodname*, der auch im Operanden LOAD-MODULE der PROGRAM- bzw. AREA-Anweisung anzugeben ist. Sie können jedoch auch LOAD-MODULE-Anweisungen generieren, denen Sie keine PROGRAM- bzw. AREA-Anweisung zuordnen (z.B. Lademodule, die Teile des Laufzeitsystems einer Programmiersprache enthalten).

Wenn die Funktionalität "Programmaustausch" (KDCAPPL PROGRAM=NEW) unter BS2000/OSD verwendet werden soll, muss mindestens eine LOAD-MODULE-Anweisung generiert werden.

Mit der Reihenfolge der LOAD-MODULE-Anweisungen und mit dem Operanden LOAD-MODE bestimmen Sie, in welcher Reihenfolge die Lademodule beim Start der UTM-Anwendung geladen werden. Nacheinander werden geladen:

- Der Basisteil der Anwendung inklusive aller Lademodule, die statisch in das Anwendungsprogramm eingebunden werden (LOAD-MODE=STATIC).
- Alle Lademodule, die beim Start der UTM-Anwendung in einen anwendungsglobalen Common Memory Pool geladen werden. Sie werden generiert mit LOAD-MODULE LOAD-MODE=(POOL, *poolname,*...) und MPOOL *poolname*,SCOPE=GLOBAL.
  Die Common Memory Pools werden in der Reihenfolge der MPOOL-Anweisungen im Generierungslauf geladen. Innerhalb eines Pools gilt die Reihenfolge der LOAD-MODULE-Anweisungen zu diesem Pool.
- Alle Lademodule, die beim Start der UTM-Anwendung in einen anwendungslokalen Common Memory Pool geladen werden. Sie werden generiert mit LOAD-MODULE LOAD-MODE=(POOL, *poolname,*...) und MPOOL *poolname*,SCOPE=GROUP.
  Die Pools werden in der Reihenfolge der MPOOL-Anweisungen geladen. Innerhalb eines Pools gilt die Reihenfolge der LOAD-MODULE-Anweisungen zu diesem Pool.

- Alle Lademodule, die beim Start als eigenständige Einheiten nachgeladen werden sollen. Sie werden generiert mit LOAD-MODULE LOAD-MODE=STARTUP. Die Lademodule werden in der Reihenfolge der so definierten LOAD-MODULE-Anweisungen geladen.

Lademodule, die mit LOAD-MODE=ONCALL generiert sind, werden beim erstmaligen Aufruf eines zugeordneten Teilprogramms geladen.

Bitte beachten Sie:

- Lademodule, die TCB-Entries enthalten, können **nicht** ausgetauscht werden!
- Es ist beim dynamischen Binden eines Lademoduls mit der Generierung ALTERNATE-LIBRARIES=YES darauf zu achten, dass tatsächlich nur RTS-Module nachgebunden werden, da bei einem Austausch der Lademodule nur das Lademodul selbst wieder aus dem Speicher entfernt wird. Wenn mit dem Lademodul per Autolink andere Module nachgeladen werden, dann verbleiben diese Module beim Austausch im Speicher, obwohl sich beispielsweise gemeinsame Datenstrukturen geändert haben.
- Für Lademodule, die nur aus C-Code und ggf. Datenobjekten (AREAs) bestehen, ist beim Binden mit der Bibliothek SYSLNK.CRTE.PARTIAL-BIND die Angabe ALTERNATE-LIBRARIES=YES nicht nötig und sollte daher vermieden werden.

```
LOAD-MODULE␣  lmodname
              [ ,ALTERNATE-LIBRARIES={ NO | YES }
              [ ,LIB=libname ]
              [ ,LOAD-MODE={ STARTUP |
                             ONCALL |
                             STATIC |
                             (POOL,poolname,{ NO-PRIVATE-SLICE |
                                              STARTUP |
                                              ONCALL } )
                             } ]
                ,VERSION=version
```

lmodname Name des Lademoduls. *lmodname* kann max. 32 Zeichen lang sein.

Für die Namensvergabe gelten die gleichen Regeln wie für Elementnamen einer Programm-Bibliothek. Siehe auch Abschnitt „Format der Namen“ auf Seite 257.

ALTERNATE-**LIB**RARIES=

Autolink-Funktion beim dynamischen Binden der Privaten Slice des Lademoduls steuern

NO Es wird kein Autolink beim Binden des Lademoduls durchgeführt.

Standard: NO

YES Die Autolink-Funktion des BLS wird eingeschaltet.
Mit dieser Funktion werden u.a. für Lademodule, die beim Austausch weitere RTS-Module benötigen, die sich noch nicht im Speicher befinden, diese RTS-Module nachgebunden. Vor dem Start der UTM-Anwendung müssen die benötigten RTS-Bibliotheken mit Linkname BLSLIB*nn* ($00 \leq nn \leq 99$) belegt werden. Diese Bibliotheken werden dann beim Nachladen der Lademodule, bei offenen Externverweisen, die sich nicht durch geladene Module befriedigen lassen, anhand *nn* sequenziell aufsteigend nach passenden Definitionen durchsucht.

ALTERNATE-LIBRARIES=YES darf nur zum Nachladen von RTS-Modulen und nicht zum Nachladen von Benutzerprogrammen verwendet werden.

Zur Autolink-Funktion siehe auch openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“.

i Die Kombination der Operandenwerte:
LOAD-MODE=STATIC / (POOL,*poolname*,NO-PRIVATE-SLICE) mit ALTERNATE-LIBRARIES=YES ist nicht zulässig und wird beim KDCDEF-Lauf abgelehnt.

LIB=libname Angabe der Programmbibliothek, aus der das Lademodul nachgeladen werden soll. *libname* kann max. 54 Zeichen lang sein.

Der Operand LIB= ist bei LOAD-MODE=STATIC ohne Bedeutung. In allen anderen Fällen muss dem Operanden LIB= ein Wert zugewiesen werden, entweder in der LOAD-MODULE-Anweisung oder in einer vorhergehenden DEFAULT-Anweisung.

LOAD-MODE=

Lademodus des Lademoduls

STARTUP Das Lademodul wird beim Start der Anwendung als eigenständige Einheit nachgeladen. Es werden Externverweise aus dem Subsystem, aus dem Klasse-3/4-Speicher und aus allen bereits geladenen Modulen der UTM-Anwendung befriedigt. Für Runtime-Systemfunktionen siehe auch den Operanden ALTERNATE-LIBRARIES=YES.

i Lademodule, die mit LOAD-MODE=STARTUP generiert werden und TCB-Entries enthalten, dürfen nicht im laufenden Betrieb ausgetauscht werden.

Standard: STARTUP

ONCALL Das Lademodul wird als eigenständige Einheit nachgeladen, wenn ein Teilprogramm oder VORGANG-Exit, das/der dem Lademodul zugeordnet ist, erstmalig aufgerufen wird. Es werden Externverweise aus dem Klasse-3/4-Speicher und aus allen bereits geladenen Modulen der UTM-Anwendung befriedigt.

Lademodule, die TCB-Entries enthalten, dürfen nicht mit LOAD-MODE=ONCALL generiert werden.

Wird mit mehreren Prozessen gearbeitet, so darf im laufenden Betrieb dieses Lademodul nicht in der Bibliothek LIB=*libname* überschrieben werden. Es könnten sonst eventuell unterschiedliche Stände des Lademoduls in einem Anwendungslauf zum Ablauf kommen.

STATIC Das Lademodul muss statisch in das Anwendungsprogramm eingebunden werden.

(POOL,poolname, NO-PRIVATE-SLICE)
Der Memory Pool wird mit einer MPOOL-Anweisung definiert.

poolname kann max. 50 Zeichen lang sein.

Das Lademodul wird beim Start der Anwendung in den Common Memory Pool *poolname* geladen. Es gibt keine Aufteilung des Lademoduls in Public und Private Slices. Es gibt also kein Private Slice, auch nicht statisch im Anwender-Programm eingebunden.

(POOL,poolname, STARTUP)
Der Public Slice des Lademoduls wird beim Start der Anwendung in den Common Memory Pool *poolname* geladen. Das zu dem Lademodul gehörige Private Slice wird anschließend in den tasklokalen Speicher geladen.

(POOL,poolname, ONCALL)
Der Public Slice des Lademoduls wird beim Start der Anwendung in den Common Memory Pool *poolname* geladen. Das zu dem Lademodul gehörige Private Slice wird in den tasklokalen Speicher geladen, wenn das erste Teilprogramm aufgerufen wird, das diesem Lademodul zugeordnet ist.

Es werden Externverweise nur aus dem Klasse-3/4-Speicher, aus den Subsystemen und aus dem eigenen Memory Pool befriedigt.

VERSION=version

Versionsnummer des Lademoduls. *version* kann max. 24 Zeichen lang sein.

Für die Vergabe der Versionsnummern gelten die gleichen Regeln wie für die Versionen von Elementen einer Programmbibliothek.

Die Angabe VERSION=@ bewirkt, dass BLS in einer PLAM-Bibliothek das Lademodul *lmodname* anspricht, das als letztes ohne explizite Versionsangabe in diese PLAM-Bibliothek gebracht wurde. Wird bei LMS mit expliziten Versionen gearbeitet, kann @ als Version eines Lademoduls **nicht** verwendet werden.

Für die Namensvergabe gelten die gleichen Regeln, wie für die Versionen von Elementen einer Programmbibliothek. Mit einer Einschränkung: wenn *version* das Zeichen „.“ enthält, so muss die Version mit einem Buchstaben beginnen.

LPAP - LPAP-Partner für die verteilte Verarbeitung über LU6.1 definieren

Mit der Steueranweisung LPAP beschreiben Sie in der lokalen Anwendung einen logischen Anschlusspunkt für die Partner-Anwendung. Eine LPAP-Anweisung ist nur erforderlich, wenn die Kommunikation mit einer Partner-Anwendung über das Protokoll LU6.1 erfolgt. Die logischen Anschlusspunkte für Partner-Anwendungen heißen LPAP-Partner. Für den LPAP-Partner legen Sie einen logischen Namen, etwaige Administrationsberechtigung der Partner-Anwendung, Maximalwerte für die Message Queue des LPAP-Partners und logische Eigenschaften für die Kommunikation mit der Partner-Anwendung über das LU6.1-Protokoll fest.

Zur Generierung von LU6.1-Verbindungen siehe auch Abschnitt „Verteilte Verarbeitung über das LU6.1-Protokoll“ auf Seite 72.

Mit der CON-Anweisung ordnen Sie dem LPAP-Partner eine reale Partner-Anwendung zu, siehe hierzu auch die CON-Anweisung auf Seite 309.

```
LPAP␣          lpapname
               [ ,BUNDLE = master_lpap_name ]
               [ ,KSET=keysetname ]
               [ ,LNETNAME=local_netname ]
               [ ,PERMIT={ ADMIN | SATADM[1] | ( ADMIN,SATADM )[1] } ]
               [ ,QLEV=queue_level_number]
               [ ,RNETNAME=remote_netname ]
                 ,SESCHA=sescha_name
               [ ,STATUS={ ON | OFF } ]
```

B *zusätzlicher Operand in BS2000/OSD*

B `[ ,NETPRIO={ MEDIUM | LOW } ]`

B [1] Nur in BS2000/OSD erlaubt

lpapname LPAP-Partnername, d.h. logischer Name der Partner-Anwendung, über den die Teilprogramme der lokalen Anwendung die Partner-Anwendung ansprechen. *lpapname* hat nur in der lokalen Anwendung eine Bedeutung und kann max. 8 Zeichen lang sein.

Der angegebene Name muss eindeutig sein und darf keinem weiteren Objekt der Namensklasse 1 zugeordnet sein. Siehe dazu auch Abschnitt „Eindeutigkeit der Namen und Adressen“ auf Seite 262.

Die LPAP-Namen bilden mit den LTERM-Namen und den OSI-LPAP-Namen eine gemeinsame Namensklasse.

BUNDLE=master_lpap_name

Name des Master-LPAP.
Durch Angabe dieses Operanden wird das LPAP zu einem Slave-LPAP eines LU6.1-LPAP-Bündels.

Den *master_lpap_name* definieren Sie mit einer MASTER-LU61-LPAP-Anweisung.

Nachrichten, die mit einem APRO-Aufruf an das Master-LPAP eines LPAP-Bündels gerichtet werden, werden von openUTM auf die Slave-LPAPs dieses LPAP-Bündels verteilt. Damit kann die Anwendung eine Verteilung der zu sendenden Nachrichten auf mehrere gleichartige Partner-Anwendungen erreichen, ohne diese Verteilung selbst programmieren zu müssen.

KSET=keysetname

Name des Keysets, das der Partner-Anwendung in der lokalen Anwendung zugeordnet wird. Das Keyset wird mit der KSET-Anweisung definiert. Die Partner-Anwendung kann nur die Services starten, bzw. nur die in der lokalen Anwendung generierten entfernten Services adressieren,

– die nicht zugriffsgesichert sind, d.h. für die kein Lockcode definiert ist,
– deren Keycodes im Keyset *keysetname* definiert sind.

Auf diese Weise kann sich die lokale Anwendung vor unberechtigten Zugriffen durch die Partner-Anwendung schützen.

Standard: kein Keyset,
d.h. es dürfen nur Transaktionscodes von der Partner-Anwendung gestartet werden, die nicht mit Lockcodes geschützt sind.

LNETNAME=local_netname

Diesen Parameter benötigen Sie nur bei heterogener Kopplung.
local_netname bezeichnet den VTAM-Namen, unter dem die UTM-Anwendung in der CICS- bzw. IMS-Partner-Anwendung bekannt ist.

Standard: Leerzeichen

B B NETPRIO= bezeichnet die Transportpriorität, die auf der diesem LPAP-Partner zugeordneten Transportverbindung benutzt werden soll.

B Standard: MEDIUM

PERMIT= legt die Berechtigungsstufe der Partner-Anwendung fest.

ADMIN Die Partner-Anwendung darf Administrationsfunktionen in der lokalen Anwendung ausführen.

B B B B SATADM Die Partner-Anwendung kann Preselection-Funktionen in der lokalen Anwendung ausführen. D.h. die Partner-Anwendung kann die SAT-Protokollierung bestimmter Ereignisse ein- bzw. ausschalten (UTM-SAT-Administrations-Berechtigung).

B B B (ADMIN,SATADM)
Die Partner-Anwendung kann Administrations- und SAT-Preselection-Funktionen in der lokalen Anwendung ausführen.

Standard:
Wenn PERMIT= nicht angegeben wird, kann die Partner-Anwendung keine Administrationsfunktionen in der lokalen Anwendung ausführen.

QLEV=queue_level_number
gibt die maximale Anzahl der asynchronen Nachrichten an, die in der Message Queue des LPAP-Partners stehen dürfen. Wird der Schwellwert überschritten, so werden weitere APRO-AM-Aufrufe an diesen LPAP-Partner mit der Meldung 40Z abgewiesen.

Standard: 32767
Minimalwert: 0
Maximalwert: 32767 (bedeutet unbeschränkt)

RNETNAME=remote_netname
Diesen Parameter benötigen Sie nur bei heterogener Kopplung. *remote_netname* bezeichnet den VTAM-Namen der CICS- bzw. IMS-Partner-Anwendung.

Standard: Leerzeichen

SESCHA=sescha_name
Unter *sescha_name* wurden in der SESCHA-Anweisung die Session-Eigenschaften für die Kommunikation zwischen der lokalen Anwendung und der Partner-Anwendung definiert, siehe hierzu auch die SESCHA-Anweisung auf Seite 474. Mit *sescha_name* können Sie dem LPAP-Partner diesen Satz Session-Eigenschaften zuordnen.

Pflichtoperand.

STATUS= legt fest, ob der LPAP-Partner gesperrt ist. Der Status kann im Betrieb mit dem Administrationskommando KDCLPAP geändert werden.

ON Der LPAP-Partner ist nicht gesperrt. Es können Verbindungen zwischen der Partner-Anwendung und der lokalen Anwendung aufgebaut werden bzw. es bestehen bereits Verbindungen.

Standard: ON

OFF Der LPAP-Partner ist gesperrt. Es können keine Verbindungen zwischen der Partner-Anwendung und der lokalen Anwendung aufgebaut werden.

LSES - Sessionnamen für die verteilte Verarbeitung über LU6.1 definieren

Die LSES-Anweisung ist nur anzugeben, wenn die Kommunikation mit der Partner-Anwendung über das Protokoll LU6.1 erfolgt.

Zur Generierung von LU6.1-Verbindungen siehe auch Abschnitt „Verteilte Verarbeitung über das LU6.1-Protokoll“ auf Seite 72.

Für die Verbindung zwischen zwei Anwendungen bei verteilter Verarbeitung werden mit der LSES-Anweisung gemeinsame Sessionnamen vereinbart. Über diese Namen wird eine unterbrochene Kommunikation wiederhergestellt. Gleichzeitig wird die Session einem LPAP-Partner zugeordnet. Jeder LPAP-Anweisung wird damit mindestens eine LSES-Anweisung zugeordnet. Bei parallelen Sessions müssen für einen LPAP-Partner *lpapname* mehrere unterschiedliche Sessionnamen definiert werden.

Einem LPAP-Partner muss immer die gleiche Anzahl an Sessions (LSES-Anweisung) und Transportverbindungen (CON-Anweisung) zugeordnet werden.

Ausnahme: In einer UTM-Cluster-Anwendung dürfen einem LPAP-Partner mehr LSES-Anweisungen als CON-Anweisungen zugeordnet werden.

Ist für die lokale Anwendung eine Session mit LSES AAA, RSES=BBB definiert, dann muss in der Partner-Anwendung diese Session mit LSES BBB, RSES=AAA generiert werden.

Damit die USER- und Sessionnamen in zwei miteinander verbundenen Anwendungen nicht eindeutig sein müssen, wird der gemeinsame Sessionname aus zwei Teilen zusammengesetzt: *sessionname* = *local_sessionname* + *remote_sessionname*

```
LSES␣          local_sessionname

                 ,LPAP=lpapname

               [ ,RSES=remote_sessionname ]
```

local_sessionname

lokaler Half-Session-Name.
Der angegebene Name muss eindeutig sein und darf auch keinem weiteren Objekt der Namensklasse 2 zugeordnet sein. Siehe dazu auch Abschnitt „Eindeutigkeit der Namen und Adressen“ auf Seite 262.

LPAP=lpapname

Name des LPAP-Partners, der der Partner-Anwendung zugeordnet wird. *local_sessionname* wird für die Kommunikation mit der Partner-Anwendung benutzt, die in der lokalen Anwendung dem LPAP-Partner *lpapname* zugeordnet ist.

RSES=remote_sessionname

remote Half-Session-Name

Standard: Wird RSES= nicht angegeben, so wird *remote_sessionname=local_sessionname* gesetzt.

LTAC - Transaktionscode für Partner-Anwendung definieren

Mit der Steueranweisung LTAC definieren Sie in der lokalen Anwendung einen Transaktionscode für einen Vorgang bzw. ein fernes Service-Programm in einer Partner-Anwendung. LTAC-Anweisungen können sowohl für die Kommunikation über das LU6.1-Protokoll als auch für die Kommunikation über das OSI TP-Protokoll generiert werden.

Dem lokalen Transaktionscode wird entweder

- (bei einstufiger Adressierung) der Name eines Transaktionscodes in einer bestimmten Partner-Anwendung zugeordnet. Dadurch adressiert der lokale Transaktionscode sowohl die Partner-Anwendung als auch den Transaktionscode in dieser Anwendung, oder
- (bei zweistufiger Adressierung) der Name eines Transaktionscodes in irgendeiner Partner-Anwendung zugeordnet. In welcher Partner-Anwendung das mit dem lokalen Transaktionscode angesprochene Service-Programm ablaufen soll, muss explizit an der Programmschnittstelle angegeben werden.

```
LTAC␣       ltacname
            [ ,{ ACCESS-LIST=keysetname | LOCK=lockcode } ]
            [ ,LPAP=lpapname ]
            [ ,LTACUNIT=ltacunit ]
            [ ,RTAC={ C'rtacname' |
                      rtacname |
                      recipient_TPSU_title [ ,CODE={ STANDARD |
                                                     PRINTABLE-STRING |
                                                     T61-STRING |
                                                     INTEGER }
                                           ]
                    }
            ]
            [ ,STATUS = { ON | OFF } ]
            [ ,TYPE={ D | A } ]
            [ ,WAITTIME=( time1,time2 ) ]
```

ltacname Name eines lokalen Transaktionscodes für das ferne Service-Programm

ACCESS-LIST=keysetname

Mit ACCESS-LIST= legen Sie die Zugriffsrechte fest, die ein Benutzer der lokalen UTM-Anwendung haben muss, um einen Auftrag an das ferne Service-Programm senden zu dürfen. Ob der Auftrag in der fernen Anwendung ausgeführt wird, hängt auch von den dort definierten Zugriffsrechten ab. ACCESS-LIST darf nicht zusammen mit dem Operanden LOCK=*lockcode* angegeben werden.

Für *keysetname* müssen Sie den Namen eines Keysets angeben. Das Keyset muss mit einer KSET-Anweisung definiert werden.

Ein Benutzer kann auf den LTAC nur dann zugreifen, wenn das Keyset des Benutzers (USER ...,KSET=), das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, und das in *keysetname* angegebene Keyset mindestens einen gemeinsamen Keycode enthalten.

Geben Sie weder ACCESS-LIST=*keysetname* noch LOCK=*lockcode* an, dann ist der LTAC nicht geschützt und jeder Benutzer der lokalen UTM-Anwendung kann das ferne Service-Programm starten.

Standard: kein Keyset

LOCK=lockcode

darf nicht zusammen mit dem Operanden ACCESS-LIST= angegeben werden.

Mit LOCK= definieren Sie den Lockcode des fernen Service-Programms. Ein mit Lockcode gesicherter Service kann von einem Teilprogramm nur dann adressiert werden, wenn das Teilprogramm unter einer Benutzerkennung (KCBENID) und von einem Client oder einer Partner-Anwendung (KCLOGTER) aus gestartet wurde, in deren Keyset ein mit dem Lockcode übereinstimmender Keycode enthalten ist.

Geben Sie weder ACCESS-LIST=*keysetname* noch LOCK=*lockcode* an, dann ist der LTAC nicht geschützt und jeder Benutzer der lokalen UTM-Anwendung kann das ferne Service-Programm starten.

Standard: 0 (kein Lockcode)
Maximalwert: Wert von MAX ...,KEYVALUE=*number*

LPAP=lpapname

gibt an, zu welcher Partner-Anwendung das Service-Programm gehört. Name des LPAP-Partners, der dieser Partner-Anwendung zugeordnet ist, oder Name eines LU6.1-LPAP-Bündels oder eines OSI-LPAP-Bündels.

Wird der Operand LPAP= nicht angegeben, so muss der Name der Partner-Anwendung im Funktionsaufruf APRO (im Feld KCPA) spezifiziert werden.

LTACUNIT=ltacunit

legt die Anzahl der Verrechnungseinheiten fest, die in der Abrechnungsphase des UTM-Accounting für jeden Aufruf dieses LTACs berechnet wird. Die Verrechnungseinheiten werden auf den Verrechnungseinheitenzähler der Benutzerkennung aufaddiert, die den LTAC aufgerufen hat.
Die Angabe ist nur ganzzahlig möglich. Dieser Operand hat nur Bedeutung, wenn mit der Funktion „UTM-Accounting“ gearbeitet wird. Siehe dazu das openUTM-Handbuch „Einsatz von openUTM-Anwendungen“.

Standardwert: 1
Minimalwert: 0
Maximalwert: 4095

RTAC= Name des Transaktionscodes für das ferne Service-Programm in der Partner-Anwendung. Mit *ltacname* wird in der lokalen Anwendung ein Service-Programm adressiert, das in der Partner-Anwendung unter dem hier angegebenen Transaktionscode (*recipient TPSU-title*) bekannt ist.

Standard: *rtacname=ltacname*

C‘rtacname‘
rtacname
recipient_TPSU_title

Der Name des Transaktionscodes für das ferne Service-Programm in der Partner-Anwendung (*recipient_TPSU_title*) kann als String oder als Zahl angegeben werden. Ein String kann in der Form *C‘rtacname‘* oder *rtacname* eingegeben werden.

Der Standard zu OSI TP unterscheidet für *recipient_TPSU_title* die Codetypen Printable String, T.61-String und Integer, die von openUTM intern für die Darstellung des RTAC-Namens verwendet werden.

CODE=**ST**AN**D**ARD

Wird *recipient_TPSU_title* als String angegeben, darf er max. 8 Zeichen lang sein und nur Zeichen enthalten, die für TAC-Namen zulässig sind. Siehe dazu auch Abschnitt „Format der Namen“ auf Seite 257.

Für die Kommunikation über das LU6.1 Protokoll muss CODE=STD verwendet werden. Wenn die Partner-Anwendung eine UTM-Anwendung ist, sollte CODE=STD verwendet werden.

Für die Kommunikation über das OSI TP-Protokoll wird intern CODE=PRINTABLE-STRING verwendet.

Standard: STANDARD

CODE=**PRINT**ABLE-STRING

Der String *recipient_TPSU_title* kann max. 64 Zeichen lang sein. Groß- und Kleinbuchstaben werden unterschieden.

Ist die Partner-Anwendung eine UTM-Anwendung, darf *recipient_TPSU_title* max. 8 Zeichen lang sein und nur Zeichen enthalten, die für TAC-Namen zulässig sind. Ein String, der diesen Anforderungen nicht entspricht, kann nur bei heterogener Kopplung über OSI TP-Protokoll verwendet werden.

Für den Codetyp PRINTABLE-STRING sind folgende Zeichen erlaubt:

– A, B, C, . . . , Z
– a, b, c, . . . , z
– 0, 1, 2, . . . , 9

und die folgenden Sonderzeichen:

Apostroph	'
Bindestrich	-
Leerzeichen	␣
Doppelpunkt	:
Fragezeichen	?
Gleichheitszeichen	=
Komma	,
Pluszeichen	+
Punkt	.
Runde Klammer auf	(
Runde Klammer zu	)
Schrägstrich	/

CODE=**T61**-STRING

Für den Codetyp T61-STRING unterstützt openUTM alle Zeichen des Codetyps PRINTABLE-STRING und zusätzlich folgende Sonderzeichen:

Dollarzeichen	$
Grösserzeichen	>
Kleinerzeichen	<
Kaufmännisches Und	&
Klammeraffe	@
Nummernzeichen	#
Prozentzeichen	%
Semikolon	;
Stern	*
Unterstrich	_

CODE=**INT**EGER

Als *recipient_TPSU_title* kann eine ganze positive Zahl zwischen 0 und 67108863 angegeben werden.

Diese Form ist nur für Partner-Anwendungen zulässig, die keine UTM-Anwendungen sind und über OSI TP-Protokoll kommunizieren.

STATUS= legt fest, ob der *ltacname* für das ferne Service-Programm nach Start der lokalen Anwendung gesperrt ist oder nicht.

Die Angabe bei STATUS= gilt so lange, bis sie mit dem Administrationskommando KDCLTAC geändert wird.

ON Der Transaktionscode *ltacname* ist nicht gesperrt, d.h. Aufträge für das entsprechende ferne Service-Programm werden angenommen.

Standard: ON

OFF Der Transaktionscode *ltacname* ist gesperrt, d.h. Aufträge für das ferne Service-Programm werden nicht angenommen.

TYPE= definiert, ob das ferne Service-Programm im Dialog oder asynchron genutzt wird.

D Das ferne Service-Programm wird im Dialog genutzt.

Standard: D

A Das ferne Service-Programm wird asynchron genutzt.

WAITTIME=(time1,time2)

Maximale Wartezeiten für Sessionbelegung festlegen. Durch geeignete Wahl dieser Wartezeiten können Sie die Wartezeit eines Benutzers am Terminal begrenzen, der den fernen Service anfordert.

time1 Zeit in Sekunden, die beim Start eines fernen Service-Programms auf das Belegen einer Session (evtl. einschließlich Verbindungsaufbau) bzw. auf den Aufbau einer Association gewartet werden soll.

- *time1* ≠ 0 bei Asynchron-TACs:
  Ein Asynchron-Auftrag wird in die Message Queue der Partner-Anwendung eingetragen.
- *time1* ≠ 0 bei Dialog-TACs:
  Ein Dialog-Auftrag wird angenommen, sofern eine logische Verbindung zur Partner-Anwendung besteht.
- *time1* = 0 bei Asynchron-TACs:
  Ein nicht-zeitgesteuerter Asynchron-Auftrag (FPUT-Auftrag) wird nur dann in die Message Queue der Partner-Anwendung eingetragen, wenn zur Partner-Anwendung eine logische Verbindung besteht. Besteht keine Verbindung wird der FPUT-Aufruf mit Returncode 40Z, KD13 abgewiesen.
- *time1* = 0 bei Dialog-TACs:
  Falls keine Session bzw. Association generiert ist, für die die lokale Anwendung Contention Winner ist, wird der Dialog-Auftrag (APRO DM-Aufruf) mit 40Z, KD11 abgewiesen. Existieren Sessions/Associations, für die die lokale Anwendung Contention Winner ist, aber beim Programmende ist keine frei, dann wird die Transaktion zurückgesetzt.

Bei Asynchron-Aufträgen an OSI TP-Partner wird *time1* intern immer auf 60 Sekunden gesetzt, unabhängig von tatsächlich eingestellten Wert.

Besteht keine logische Verbindung zur Partner-Anwendung, dann werden Dialog-Aufträge abgewiesen, unabhängig davon, welchen Wert *time1* hat. Gleichzeitig wird ein Verbindungsaufbau initiiert.

time2 Maximale Zeit in Sekunden, die auf das Eintreffen einer Antwort vom Auftragnehmer gewartet wird.
time2 = 0 bedeutet „Warten ohne Zeitbegrenzung“.
time2 ist nur für Dialog-LTACs relevant, die Wartezeiten für Asynchron-LTACs werden mit UTMD ... CONCTIME=(...,*time2*) festgelegt.

Standardwert: WAITTIME = (30,0).
Minimalwert: WAITTIME = (0,0)
Maximalwert: WAITTIME = (32767,32767)

Die Wartezeiten können durch die UTM-Administration geändert werden (z.B. mit dem Kommando KDCLTAC).

LTERM - LTERM-Partner für Clients und Drucker definieren

Mit der Steueranweisung LTERM definieren Sie LTERM-Partner als logische Anschlusspunkte für die Clients und Drucker der Anwendung, d.h. für Terminals, UPIC-Clients und Transportsystem-Anwendungen (DCAM-, PDN-, CMX-, Socket-Anwendungen oder UTM-Anwendungen, die als Transportsystem-Anwendung generiert sind).

Über den LTERM-Partner bauen die Clients oder Drucker die Verbindung zur UTM-Anwendung auf. Den LTERM-Partnern werden mit der PTERM-Anweisung anschließend physische Clients oder Drucker zugewiesen. Sie können auch Pools von LTERM-Partnern definieren, siehe dazu die TPOOL-Anweisung auf Seite 518.

LTERM-Partner können Sie auch vordefinieren, d.h. sie noch keinem Client/Drucker zuordnen. Die Zuweisung LTERM-Partner → PTERM kann später im Betrieb mit dem Administrationskommando KDCSWTCH erfolgen.

Für alle Clients, die mit einer PTERM-Anweisung definiert werden, muss jeweils eine eigene LTERM-Anweisung geschrieben werden.

W Drucker werden von openUTM unter Windows-Systemen nicht unterstützt.

```
LTERM␣      ltermname
            [ ,BUNDLE=master-lterm]
            [ ,GROUP=primary-lterm]
            [ ,KSET=keysetname ]
            [ ,LOCK=lockcode ]
            [ ,QAMSG={ YES | NO } ]
            [ ,QLEV=queue_level_number ]
            [ ,RESTART={ YES | NO } ]
            [ ,STATUS={ ON | OFF } ]
            [ ,USAGE={ D | O } ]
            [ ,USER=username ]

B/X         BS2000/OSD- und Unix-System-spezifische Operanden
B/X         [ ,CTERM=ltermname2 ]
B/X         [ ,PLEV=print_level_number ]

B           BS2000/OSD-spezifische Operanden
B           [ ,ANNOAMSG={ Y | N } ]
B           [ ,FORMAT= { + | * | # }formatname ]
B           [ ,KERBEROS-DIALOG = { YES | NO } ]
B           [ ,LOCALE=( [ lang_id ][,[ terr_id ][ ,ccsname ] ] ) ]
B           [ ,NETPRIO={ MEDIUM | LOW } ]
```

Die Operanden LOCK=, KSET=, USER= und ANNOAMSG= gelten nur für Clients; die Operanden CTERM= und PLEV= nur für Drucker.

ltermname Name des LTERM-Partners. *ltermname* kann max. 8 Zeichen lang sein.

Über *ltermname*

- wird dem LTERM-Partner in der PTERM-Anweisung ein Client oder Drucker zugewiesen.
- sprechen die Teilprogramme der Anwendung die Clients, Drucker und andere TS-Anwendungen (keine Server-Server-Kommunikation) an, die dem LTERM-Partner zugeordnet sind.

Der angegebene Name muss eindeutig sein und darf keinem weiteren Objekt der Namensklasse 1 zugeordnet sein. Siehe dazu auch Abschnitt „Eindeutigkeit der Namen und Adressen“ auf Seite 262.

B B B ANNOAMSG= (**anno**unce **a**synchronous **m**e**s**sa**g**e)
Gilt nur für LTERM-Partner, über die sich Terminals (USAGE=D) an die UTM-Anwendung anschließen.

B B B Y Eine asynchrone Nachricht an dieses Terminal soll mit der Meldung K012 in der Systemzeile angekündigt werden. Der Benutzer muss die Nachricht mit dem UTM-Benutzerkommando KDCOUT anfordern.

B Standard: Y

B B B B N Eine asynchrone Nachricht an das Terminal wird sofort, d.h. ohne Ankündigung, gesendet. Wenn ANNOAMSG=N generiert ist, kann zu diesem Terminal erst ab OMNIS V7.0 ein Verbindungsaufbau über eine Multiplexverbindung (siehe MUX-Anweisung auf Seite 417) erfolgen.

BUNDLE=master-lterm
Name eines Master-LTERM in einem LTERM-Bündel (Verbindungsbündel). Durch die Angabe von *master-lterm* wird dieses LTERM zum Slave-LTERM des zugehörigen Verbindungsbündels.

Das Master-LTERM, das hier angegeben wird, muss in einer vorangegangenen LTERM-Anweisung generiert worden sein. Dem Master-LTERM darf kein PTERM zugeordnet sein.

Verbindungsbündel ermöglichen eine Lastverteilung (siehe Abschnitt „LTERM-Bündel“ auf Seite 133).

Verbindungsbündel können für APPLI- oder SOCKET-Verbindungen (Operand PTYPE der zugehörigen PTERM-Anweisung) generiert werden.

BUNDLE darf nicht zusammen mit GROUP oder CTERM angegeben werden.

B/X CTERM=ltermname2

B/X (**c**ontrol **term**inal)

B/X Wird nur für LTERM-Partner angegeben, die für Drucker (USAGE=O) generiert werden.

B/X *ltermname2* ist der max. 8 Zeichen lange Name eines LTERM-Partners, der

B/X als Druckersteuer-LTERM konfiguriert wurde (LTERM ...,USAGE=D). Dem

B/X Druckersteuer-LTERM können ein oder mehrere LTERM-Partner, die für

B/X Drucker konfiguriert wurden, zugeordnet werden. Über das Druckersteuer-

B/X LTERM können Drucker, Druckaufträge und die Drucker Queues administ-

B/X riert werden.

B/X Standard: Leerzeichen, d.h. kein Druckersteuer-LTERM

GROUP=primary-lterm

Name eines Primary-LTERMs. Durch die Angabe von *primary-lterm* wird dieses LTERM zu einem Alias-LTERM des zugehörigen Primary-LTERM. Sie definieren eine LTERM-Gruppe.

In einer LTERM-Gruppe ordnen Sie mehrere LTERMs einer Verbindung zu (siehe Abschnitt „LTERM-Gruppen“ auf Seite 136).

Das Primary-LTERM, das hier angegeben wird, muss in einer vorangegangenen LTERM-Anweisung generiert worden sein. Das Primary-LTERM muss entweder ein normales LTERM, das einem PTERM mit PTYPE=APPLI oder PTYPE=SOCKET zugeordnet ist, oder das Master-LTERM eines Verbindungsbündels sein. Einem Alias-LTERM darf kein PTERM zugeordnet sein.

GROUP darf nicht zusammen mit BUNDLE oder CTERM angegeben werden.

FORMAT= Bezeichnet das Startformat des LTERM-Partners. Ein Startformat kann nur für Terminals definiert werden. Die Angabe eines Startformats ist nur dann sinnvoll, wenn die Anwendung ohne Benutzerkennungen generiert oder wenn ein eigener Anmelde-Vorgang eingesetzt wird.

Wenn der LTERM-Partner einem UPIC-Client zugeordnet wird, ist die Angabe eines Startformates wirkungslos.

Ist die Anwendung **ohne** Benutzerkennungen generiert, dann wird dieses Format an Stelle der Meldung K001 ausgegeben. Findet ein Terminal-spezifischer Wiederanlauf statt, so erscheint **nicht** das Startformat, sondern es erfolgt ein KDCDISP.

Ist die Anwendung **mit** Benutzerkennungen generiert, kann der Name des Startformates im ersten Teil des Anmelde-Vorgangs mit einem SIGN ST-Aufruf abgefragt werden. Wird kein eigener Anmelde-Vorgang eingesetzt, lässt sich das LTERM-spezifische Startformat nicht nutzen.

Das Formatkennzeichen setzt sich wie folgt zusammen:

+, * oder # gefolgt von einem max. 7 Zeichen langen alphanumerischen Namen (*formatname*).

#Formate können nur im Zusammenhang mit einem Anmelde-Vorgang genutzt werden.

Es bedeuten:

+ Beim nächsten MGET-Aufruf des Teilprogramms werden im Nachrichtenbereich (NB) jedem Inhalt eines Formatfeldes 2 Byte für das Attributfeld vorangestellt, d.h. die Feldeigenschaften sind vom Teilprogramm veränderbar. Der Formatname an der KDCS-Schnittstelle ist +*formatname*.

* Dem Inhalt eines Formatfeldes wird im Nachrichtenbereich des nächsten MGET-Aufrufs des Teilprogramms kein Byte für ein Attributfeld vorangestellt, d.h. die Feldeigenschaften sind vom Teilprogramm nicht veränderbar. Der Formatname an der KDCS-Schnittstelle ist **formatname*.

bezeichnet ein Format mit erweiterten Benutzerattributen (**Extend**ed User Attributes). Feldeigenschaften und globale Formateigenschaften sind vom Teilprogramm veränderbar. Der Formatname an der KDCS-Schnittstelle ist #*formatname*.

Standard: kein Startformat

B KERBEROS-DIALOG =

B B B YES Beim Verbindungsaufbau wird für Terminals, die Kerberos unterstützen und die sich direkt (nicht über OMNIS) über diesen LTERM-Partner an die Anwendung anschließen, ein Kerberos-Dialog durchgeführt.

B B B B openUTM speichert die Kerberos-Information in der Länge ab, die sich aus dem Maximum der bei MAX PRINCIPAL-LTH und MAX CARDLTH generierten Längen ergibt. Wenn die Kerberos-Information länger ist, wird sie auf diese Länge verkürzt abgespeichert.

B B B B Mit dem KDCS-Aufruf INFO (KCOM=CD) kann ein Teilprogrammlauf diese Information lesen, außer an diesem Client hat sich anschließend ein Benutzer mit Ausweiskarte angemeldet, da in diesem Fall die Kerberos-Information durch die Ausweis-Information überschrieben worden ist.

B B B Wenn das Maximum, das sich aus den bei MAX PRINCIPAL-LTH und MAX CARDLTH generierten Längen ergibt, gleich Null ist, wird eine Warnungsmeldung ausgegeben.

B NO Es wird kein Kerberos-Dialog durchgeführt.

B Standard.

KSET=keysetname

Gilt nur für Clients, die als Dialog-Partner generiert sind (USAGE=D). *keysetname* ist der Name eines Keysets, das mit der KSET-Anweisung definiert und dem LTERM-Partner *ltermname* zugeordnet wird. *keysetname* darf bis zu 8 Zeichen lang sein.

Jedem LTERM-Partner kann maximal ein Keyset zugeordnet werden. Es legt die Zugriffsrechte dieses LTERM-Partners für die Nutzung von Services der Anwendung und von entfernten Services (LTAC) fest, die in dieser Anwendung generiert wurden.

Über diesen LTERM-Partner können mit Lockcode bzw. Access-List geschützte Services der Anwendung nur gestartet bzw. mit Lockcode bzw. Access-List geschützte entfernte Services nur adressiert werden, wenn Folgendes gilt: Sowohl im zugeordneten Keyset des LTERM-Partners als auch im KSET der UTM-Benutzerkennung, unter der die Anmeldung über diesen LTERM-Partner erfolgte, ist der zum Lockcode bzw. der Access-List passende Key- bzw. Zugangscode enthalten.
Das Lock-/Keycode- und das Access-List-Konzept werden ausführlich im openUTM-Handbuch „Konzepte und Funktionen“ beschrieben. Eine Einführung in die Zugriffskontrolle finden Sie im gleichnamigen Abschnitt „Lock-/Keycode-Konzept“ auf Seite 216.

Services, deren TACs nicht mit Codes gesichert sind, kann der Benutzer bzw. das Client-Programm ohne Einschränkung aufrufen.

Bei einer Anwendung, in der Benutzerkennungen definiert sind und für die Sie keinen Zugriffsschutz für die Terminals definieren wollen, können Sie den Terminals alle Keycodes vergeben mit:

```
LTERM␣...,KSET=MASTERSET
KSET␣MASTERSET,KEYS=MASTER
```

Standard: Kein Keyset

B LOCALE=(lang_id,terr_id,ccs_name)
B definiert die Sprachumgebung des Clients, der sich über diesen LTERM-
B Partner an die Anwendung anschließt.

B lang_id Max. zwei Zeichen langes Sprachkennzeichen des Clients. Das Sprach-
B kennzeichen ist frei wählbar.

B Das Sprachkennzeichen kann von den Teilprogrammen der Anwendung
B abgefragt werden, so dass es Nachrichten an das Terminal in der Landes-
B sprache des Kommunikationspartners übertragen kann.

B terr_id Max. zwei Zeichen langes Territorialkennzeichen für den Client. Das
B Territorialkennzeichen ist frei wählbar.

B Das Territorialkennzeichen kann von den Teilprogrammen der Anwendung
B abgefragt werden. So können in Nachrichten an das Terminal die territoria-
B len Besonderheiten in der Landessprache des Kommunikationspartners
B berücksichtigt werden.

B ccsname (**c**oded **c**haracter **s**et **name**)
B Für *ccsname* ist der max. 8 Zeichen lange Name eines erweiterten Zeichen-
B satzes (CCS-Name) anzugeben. Der angegebene CCS-Name muss zu
B einem im BS2000-System definierten EBCDIC-Zeichensatz gehören (siehe
B auch Benutzerhandbuch zu XHCS). Der Zeichensatz muss kompatibel sein
B zu einem ISO-Zeichensatz, der von dem Terminal unterstützt wird, das
B diesem LTERM-Partner zugeordnet ist.

B Zum Zeitpunkt der Generierung kann KDCDEF weder die Gültigkeit des
B CCS-Namens im BS2000-System noch die Kompatibilitätsbedingung über-
B prüfen. Wird ein CCS-Name angegeben, der in XHCS nicht definiert ist, so
B führt das im Betrieb der Anwendung beim Verbindungsaufbau über diesen
B LTERM-Partner zum PEND ER.

B Der zum CCS-Name gehörende Zeichensatz wird verwendet für:

B – die Ausgabe von Dialog-Nachrichten an 8-Bit-Terminals, wenn die
B Anwendung ohne Benutzerkennungen generiert oder wenn noch kein
B Benutzer am Terminal angemeldet ist, und wenn nicht explizit ein ande-
B rer CCS-Name über Editprofil oder Format ausgewählt wird.

B – die Ausgabe asynchroner Nachrichten an 8-Bit-Terminals, wenn nicht explizit ein anderer CCS-Name über Editprofil oder Format ausgewählt wird.

B Standard: Das in der MAX-Anweisung definierte Locale.

LOCK=lockcode

Gilt nur für Clients (USAGE=D).
Lockcode, der dem LTERM-Partner als logisches Zahlenschloss zugeordnet ist. *lockcode* ist ein Zahlenwert zwischen 1 und dem in der Anwendung erlaubten Maximalwert (MAX ...,KEYVALUE=*number*). An diesem LTERM-Partner ist eine Anmeldung nur unter einer UTM-Benutzerkennung (USER) möglich, für die ein Keyset mit einem Keycode generiert wurde, der den Lockcode des LTERM-Partners enthält.

Wenn für die Anwendung keine Benutzerkennungen generiert sind (keine USER-Anweisung), wird der Operand LOCK= ignoriert.

Standard: 0, d.h. kein Lockcode
Maximalwert: Der Wert von MAX ...,KEYVALUE=*number*

B NETPRIO= bezeichnet die Transportpriorität, die auf der Transportverbindung benutzt werden soll, die diesem LTERM-Partner zugeordnet ist.

B NETPRIO hat keine Bedeutung für LTERM-Partner, die einem PTERM mit PTYPE=SOCKET oder PTYPE=*RSO zugeordnet sind.

B Standard:
B MEDIUM für Clients
B LOW für Drucker

B/X PLEV=print_level_number

B/X (**p**rint **lev**el)
B/X Gilt nur für Drucker. Der Operand PLEV= erleichtert es dem Benutzer Drucker von verschiedenen UTM-Anwendungen aus zu benutzen (Drucker Sharing). Dazu wird die Verbindung zwischen einer UTM-Anwendung und dem Drucker nur für die Dauer der Übermittlung des Druckauftrags aufrechterhalten, um anderen Anwendungen die Möglichkeit zum Verbindungsaufbau zu geben.

B/X Der Operand legt die Anzahl der Drucker-Nachrichten fest, bei der
B/X openUTM versucht, die Verbindung zu diesem Drucker aufzubauen.
B/X openUTM sammelt die Nachrichten für einen Drucker so lange, bis der mit
B/X PLEV= definierte Schwellwert erreicht wird. Dann baut openUTM die logi-
B/X sche Verbindung zu diesem Drucker auf. Die Verbindung wird wieder abge-
B/X baut, wenn keine Nachrichten für diesen Drucker mehr vorhanden sind.
B/X Dann kann ggf. eine andere Anwendung Nachrichten auf diesem Drucker
B/X ausgeben.

B/X B/X B/X B/X Ist dem Client ein Druckerbündel zugeordnet, so versucht openUTM bei Erreichen des Schwellwerts den Verbindungsaufbau für alle Drucker dieses Bündels. Ist keine Nachricht mehr da, wird die Verbindung zu allen Druckern des Bündels abgebaut.

B/X B/X B/X B/X Wird die Verbindung zu dem Drucker abgebaut, obwohl der Schwellwert PLEV= noch überschritten ist (z.B. Abbau durch Administration), so versucht openUTM im Abstand des von MAX ...,CONRTIME=*time* definierten Zeitraums erneut, die Verbindung aufzubauen.

B/X B/X Bei PLEV=0 wird die Verbindung nicht abgebaut, wenn keine Ausgabe-Nachrichten mehr vorliegen.

B/X B/X B/X Bei PLEV>0 dürfen für den LTERM-Partner die Operanden RESTART=NO oder USAGE=D nicht angegeben werden. In BS2000/OSD darf darüber hinaus QAMSG=NO nicht angegeben werden.

B/X B/X Der Operand CONNECT=YES der zugehörigen PTERM-Anweisung hat bei PLEV> 0 keine Wirkung.

B/X B/X B/X B/X B/X Standardwert: 0
Minimalwert: 0
Maximalwert: 32767
Bei Überschreitung des Maximalwertes wird von KDCDEF ohne Meldung der Standardwert eingesetzt.

QAMSG= (**q**ueue **a**synchronous **m**e**s**sa**g**e)

YES Eine asynchrone Nachricht (FPUT-Auftrag) an den Client oder Drucker, soll von openUTM in der Message Queue dieses LTERM-Partners zwischengespeichert werden, auch wenn der Client oder Drucker nicht mit der Anwendung verbunden ist.

Standard: bei RESTART=YES

NO Ein FPUT-Auftrag an diesen Client oder Drucker wird mit den Returncodes KCRCCC=44Z und KCRCDC=K705 abgewiesen, falls der Client oder Drucker nicht mit der Anwendung verbunden ist.

Standard: bei RESTART=NO

QLEV=queue_level_number
(**q**ueue **lev**el)
legt die maximale Anzahl asynchroner Nachrichten fest, die openUTM in der Message Queue des LTERM-Partners gleichzeitig zwischenspeichert. Wird dieser Schwellwert überschritten, so weist openUTM weitere FPUT- oder DPUT-Aufrufe für diesen LTERM-Partner mit 40Z ab.

Mit QLEV= kann die Größe des Pagepools besser kontrolliert werden, weil die Anzahl der asynchronen Nachrichten nicht größer als maximal *queue_level_number* werden kann.

Standardwert: 32767
Minimalwert: 0
Maximalwert: 32767 (bedeutet unbeschränkt)
Bei Überschreitung des Maximalwertes wird von KDCDEF ohne Meldung der Standardwert eingesetzt.

Die Angabe von QLEV < PLEV für einen Drucker ist nicht sinnvoll, da dann die Verbindung zu diesem Drucker immer per Administration aufgebaut werden muss.

RESTART= Behandlung von Asynchron-Nachrichten bei Verbindungsabbau zum Client

YES Beim Verbindungsabbau zum Client, der diesem LTERM-Partner zugeordnet ist, bleiben asynchrone Nachrichten erhalten. Sind in dieser Anwendung keine Benutzerkennungen generiert, so führt openUTM für diesen LTERM-Partner den automatischen Vorgangswiederanlauf durch.

Standard für Drucker (LTERM mit USAGE=O) sowohl in stand-alone UTM-Anwendungen als auch in UTM-Cluster-Anwendungen: YES

Standard für Dialog-LTERM (LTERM mit USAGE=D) in stand-alone Anwendungen: YES

NO Beim Verbindungsabbau zum Client, der diesem LTERM-Partner zugeordnet ist, löscht openUTM alle asynchronen Nachrichten in der Message-Queue des LTERM-Partners. Handelt es sich dabei um Nachrichten eines Meldungs-Komplexes, so wird der negative Quittungsauftrag aktiviert. Durch die Angabe von RESTART=NO kann der Pagepool entlastet werden.

Bei QAMSG=YES darf RESTART=NO nicht angegeben werden.

Sind für die Anwendung keine Benutzerkennungen definiert, so führt openUTM für den Client oder Drucker keinen automatischen Vorgangswiederanlauf durch, d.h.:

– Wird die Verbindung durch KDCOFF bzw. Verbindungsverlust abgebaut oder die Anwendung normal beendet, so wird der Vorgang auf den letzten Sicherungspunkt zurückgesetzt, beendet und danach der Event-Exit VORGANG mit KCKNZVG=D (=Disconnect) aufgerufen.

– Nach einer abnormalen Beendigung der Anwendung wird beim UTM-Warmstart ein noch offener Vorgang für diesen LTERM-Partner beendet, ohne dass der Event-Exit VORGANG aufgerufen wird.

– KDCDISP/KDCLAST zeigt nach Verbindungsaufbau das gleiche Verhalten wie nach einer Neugenerierung, d.h. die Meldung `K020 KEINE NACHRICHT VORHANDEN` wird ausgegeben.

Standard für Dialog-LTERM (LTERM mit USAGE=D) in UTM-Cluster-Anwendungen: NO

STATUS= Status des LTERM-Partners nach Verbindungsaufbau. Der Status kann im Anwendungsbetrieb mit dem Administrationskommando KDCLTERM geändert werden.

ON Der Client oder Drucker, der diesem LTERM-Partner zugeordnet ist, ist nicht gesperrt. D.h. Sie können nach dem Verbindungsaufbau sofort arbeiten.

Standard: ON

OFF Der Client oder Drucker, der diesem LTERM-Partner zugeordnet ist, ist gesperrt.

USAGE= Art des LTERM-Partners.

D Der LTERM-Partner ist als Dialog-Partner konfiguriert. Auf den Verbindungen zwischen dem Client und der lokalen Anwendung können sowohl der Client als auch die Anwendung Nachrichten senden.

Standard: D

O Der LTERM-Partner ist für ein Ausgabemedium konfiguriert. Es können nur Nachrichten von der Anwendung zum Drucker oder TS-Anwendung etc. gesendet werden.

USER=username

Gilt nur, wenn der LTERM-Partner als Dialog-Partner konfiguriert wurde (USAGE=D). Je nach Typ des zugeordneten Client wirkt dieser Operand wie folgt:

– Wenn dem LTERM-Partner ein Terminal zugeordnet ist, dann führt openUTM für die Benutzerkennung *username* beim Aufbau der logischen Verbindung zwischen dem Client, der diesem LTERM-Partner zugeordnet ist, und der UTM-Anwendung ein automatisches KDCSIGN durch (=automatischer Berechtigungsnachweis). Beachten Sie, dass bei Verwendung des automatischen KDCSIGN der Zugriffsschutz eingeschränkt ist. Geben Sie diesen Operanden nur an, wenn sichergestellt ist, dass an diesem Client nur ein dazu berechtigter Benutzer unter dieser Benutzerkennung arbeitet. Nach dem Aufbau der logischen Verbindung hat der Client den Zustand, als ob der Benutzer *username* das

KDCSIGN-Kommando (BS2000/OSD) bzw. die Berechtigungsprüfung (Unix-, Windows-Systeme) erfolgreich ausgeführt hätte. Die Benutzerkennung muss mit der USER-Anweisung definiert sein.

– Wenn dem LTERM-Partner ein Client vom Typ APPLI, SOCKET oder UPIC zugeordnet wird, dann ist die Benutzerkennung *username* für diesen Client reserviert (als Verbindungs-Benutzerkennung). Der Client wird beim Aufbau der Verbindung unter dieser Benutzerkennung angemeldet. Ein anderer Client oder Terminal-Benutzer kann sich nicht mit dieser Benutzerkennung bei der UTM-Anwendung anmelden.

  Wurde über eine USER-Anweisung explizit eine Benutzerkennung mit dem Namen des LTERM-Partners generiert, dann ordnet openUTM diese Benutzerkennung dem LTERM-Partner exklusiv zu.

  Schickt der LTERM-Partner Administrationsaufrufe an die Anwendung, dann muss eine Benutzerkennung mit Administrationsberechtigung angegeben werden, falls sich der Client nicht unter einer echten Benutzerkennung anmeldet.

  Sollen von diesem Client Transaktionscodes aufgerufen werden, die mit Lockcodes geschützt sind, dann muss dieser Benutzerkennung ein passendes Keyset zugewiesen werden.

  Wurde keine Benutzerkennung angegeben, dann erzeugt KDCDEF implizit eine Benutzerkennung mit dem Namen des LTERM-Partners und dem bei LTERM ...,RESTART= definierten Wert.

Standard: kein automatisches KDCSIGN

MASTER-LU61-LPAP – Master-LPAP eines LU6.1-LPAP-Bündels definieren

Mit der Steueranweisung MASTER-LU61-LPAP legen Sie den Namen und die Eigenschaften eines Master-LPAPs für ein LU6.1-LPAP-Bündel fest.

Einem Master-LPAP eines LU6.1-LPAP-Bündels werden mit dem Parameter BUNDLE der LPAP-Anweisung Slave-LPAPs zugeordnet. Das Master-LPAP und die Slave-LPAPs bilden zusammen ein LPAP-Bündel. LPAP-Bündel ermöglichen eine automatische Verteilung von Nachrichten auf mehrere LPAP-Partner (siehe Abschnitt „LU6.1-LPAP-Bündel“ auf Seite 83).

```
MASTER-LU61-LPAP   master_lpap_name
                   [ ,STATUS={ ON | OFF } ]
```

master_lpap_name

Name für das Master-LPAP eines LU6.1-LPAP-Bündels. Dieser Name ist nur in der lokalen Anwendung von Bedeutung und muss sich von den in dieser Anwendung definierten Namen von LTERMs, LPAPs, OSI-LPAPs und TACs unterscheiden.

master_lpap_name kann maximal 8 Zeichen lang sein.

STATUS = legt fest, ob das MASTER-LU61-LPAP gesperrt ist.

- ON — Das MASTER-LU61-LPAP ist nicht gesperrt.
- OFF — Das MASTER-LU61-LPAP ist gesperrt. Aufträge für das MASTER-LU61-LPAP werden abgelehnt.

MASTER-OSI-LPAP - Master-LPAP eines OSI-LPAP-Bündels definieren

Mit der Steueranweisung MASTER-OSI-LPAP legen Sie den Namen und die Eigenschaften eines Master-LPAPs für ein OSI-LPAP-Bündel fest.

Einem Master-LPAP eines OSI-LPAP-Bündels werden mit dem Parameter BUNDLE der Anweisung OSI-LPAP Slave-LPAPs zugeordnet. Das Master-LPAP und die Slave-LPAPs bilden zusammen ein LPAP-Bündel. LPAP-Bündel ermöglichen eine automatische Verteilung von Nachrichten auf mehrere LPAP-Partner (siehe Abschnitt „OSI-LPAP-Bündel“ auf Seite 100).

```
MASTER-OSI-LPAP  master_lpap_name
                   ,APPLICATION-CONTEXT=context_name
                 [ ,STATUS={ ON | OFF }]
```

master_lpap_name
: Name für das Master-LPAP in einem OSI-LPAP-Bündel. Dieser Name ist nur in der lokalen Anwendung von Bedeutung. Er muss sich von den in dieser Anwendung definierten Namen von LTERMs, LPAPs, OSI-LPAPs und TACs unterscheiden.
master_lpap_name kann maximal 8 Zeichen lang sein.

APPLICATION-CONTEXT=context_name
: Name des Application Context, der für die Kommunikation mit dem entfernten Partner genutzt werden soll.

  Allen Slave-LPAPs eines OSI-LPAP-Bündels muss der gleiche Application Context zugeordnet sein wie dem Master-LPAP.

STATUS=
: legt fest, ob das MASTER-OSI-LPAP gesperrt ist.

  ON
  : Das MASTER-OSI-LPAP ist nicht gesperrt.

  OFF
  : Das MASTER-OSI-LPAP ist gesperrt. Aufträge für das MASTER-OSI-LPAP werden abgelehnt.

MAX - UTM-Anwendungsparameter definieren

Mit der Steueranweisung MAX legen Sie die Maximalwerte, Timer, Prozesszahlen und Systemparameter einer UTM-Anwendung fest, wie beispielsweise:

- Name der Anwendung
- Basisname bzw. Basisverzeichnis für UTM-Dateien
- einfache oder doppelte KDCFILE-Führung
- Größe einer UTM-Seite (Blockgröße für UTM-Speicher und -Puffer)
- Größe von Pagepool, Wiederanlaufbereich, Cache etc.
- Maximalwerte für Anzahl der
  - Prozesse
  - Keycodes
  - GSSBs
  - LSSBs
  - UTM-Seiten im Puffer für Benutzer-Protokollsätze etc.
- Schwellwert für die Größenüberwachung der SYSLOG-Dateigenerationen, wenn die SYSLOG als FGG angelegt ist (Operand SYSLOG-SIZE)

B
- Standard-Sprachumgebung der UTM-Anwendung (Operand LOCALE)
- ob SM2 zur Überwachung der Performance in der Anwendung eingesetzt werden kann

X
- Art der Netzanbindung, d.h. multi- oder single-threaded.

Die Parameter der Steueranweisung MAX können Sie auf mehrere MAX-Anweisungen aufteilen. Ist der gleiche Operand versehentlich in mehreren MAX-Anweisungen angegeben, wird die **erste** Angabe als gültig angenommen.

Pflichtoperanden:

APPLINAME=, KDCFILE= und TASKS=

X/W *Zusätzliche Pflichtoperanden unter Unix- und Windows-Systemen:*

X/W SEMKEY= bzw. SEMARRAY= (Semaphorschlüssel), IPCSHMKEY=, KAASHMKEY= und
X/W CACHESHMKEY=

X/W In OSI TP-Anwendungen zusätzlich: XAPTPSHMKEY= und OSISHMKEY=.

Jeder Pflichtoperand muss genau einmal angegeben werden.

Im Anschluss an die Beschreibung der Operanden dieser Anweisung finden Sie noch einmal alle Operanden der MAX-Anweisung übersichtlich in einer Tabelle zusammengefasst.

```
                    betriebssystemunabhängige Operanden
MAX␣            [ APPLIMODE={ SECURE | FAST } ]
                  ,APPLINAME=appliname
                [ ,ASYNTASKS={ atask_number | (atask_number,service_number) } ]
                [ ,BLKSIZE={ 2K | 4K } ]
                [ ,CACHESIZE=( number,paging,{ NORES | RES }[1] ) ]
                [ ,CLRCH={ c | C'c'| X'xx' } ]
                [ ,CONN-USERS=number ] (Pflichtoperand in Unix-/Windows-Systemen)
                [ ,CONRTIME=time ]
                [ ,DEAD-LETTER-Q-ALARM=number ]
                [ ,DESTADM=destination ]
                [ ,DPUTLIMIT1=( day,hour,minute,second ) ]
                [ ,DPUTLIMIT2=( day,hour,minute,second ) ]
                [ ,GSSBS=number ]
                [ ,HOSTNAME=name ]
                [ ,KB=length ]
                  ,KDCFILE=( filebase [, { SINGLE | DOUBLE } ] )
                [ ,KEYVALUE=number ]
                [ ,LEADING-SPACES={ NO | YES } ]
                [ ,LPUTBUF=number ]
                [ ,LPUTLTH=length ]
                [ ,LSSBS=number ]
                [ ,NB=length ]
                [ ,NRCONV=number ]
                [ ,OSI-SCRATCH-AREA=value ]
                [ ,PGPOOL=( number,warnlevel1,warnlevel2 ) ]
                [ ,PGPOOLFS=number ]
```

B [1] NORES | RES nur in BS2000/OSD

Fortsetzung:

```
MAX␣        [ ,PGWTTIME=time ]
            [ ,QTIME=( qtime1,qtime2 ) ]
            [ ,RECBUF=( number,length ) ]
            [ ,RECBUFFS=number ]
            [ ,REDELIVERY=(number1, number2) ]
            [ ,RESWAIT={ time1 | ( time1, time2 ) } ]
            [ ,SM2={ NO | OFF | ON } ]
            [ ,SPAB=length ]
            [ ,STATISTICS-MSG={ NONE | FULL-HOUR } ]
            [ ,SYSLOG-SIZE=size ]
              ,TASKS=number
            [ ,TASKS-IN-PGWT=number ]
            [ ,TERMWAIT=time ]
            [ ,TRACEREC=number ]
            [ ,TRMSGLTH=length ]
            [ ,USLOG={ SINGLE | DOUBLE } ]
```

B ***zusätzliche Operanden in BS2000/OSD***

```
B           [ ,BRETRYNR=number ]
B           [ ,CARDLTH=length ]
B           [ ,CATID=( catalog_A,catalog_B ) ]
B           [ ,LOCALE=( [ lang_id ][,[ terr_id ][ ,ccsname ] ] ) ]
B           [ ,LOGACKWAIT=time ]
B           [ ,MP-WAIT=number ]
B           [ ,PRINCIPAL-LTH=length ]
B           [ ,REQNR=number ]
B           [ ,SAT={ ON | OFF } ] ]
B           [ ,VGMSIZE=number ]
```

Fortsetzung:

```
X/W                  zusätzliche Operanden auf Unix-  und Windows-Systemen
X/W  MAX␣              ,CACHESHMKEY=number
X/W                    ,IPCSHMKEY=number
X/W                  [ ,IPCTRACE=number ]
X/W                    ,KAASHMKEY=number
X/W                  [ ,NET-ACCESS={ MULTI-THREADED | SINGLE-THREADED } ]
X/W                    ,OSISHMKEY=number          nur Pflicht, wenn Sie OSI TP-Partner generieren
X/W                    ,{ SEMARRAY=( number,number1 ) | SEMKEY=( number,...) }
X/W                    ,XAPTPSHMKEY=number        nur Pflicht, wenn Sie OSI TP-Partner generieren
```

APPLIMODE= legt fest, ob es sich um eine UTM-S- oder UTM-F-Anwendung handelt.

SECURE Die Anwendung wird als eine UTM-S-Anwendung generiert.

Bei UTM-S sichert openUTM alle Benutzerdaten auch über ein Anwendungsende und einen Systemausfall hinaus. UTM-S garantiert bei allen Störungen die Sicherheit und Konsistenz der Anwendungsdaten. Bei abnormalem Ende einer UTM-S-Anwendung findet beim nachfolgenden Start ein automatischer Restart statt. Zu diesem Zweck sichert openUTM bei dieser Variante am Transaktionsende alle Änderungen.

Standard: SECURE

FAST Die Anwendung wird als eine UTM-F-Anwendung generiert.

Bei UTM-F wird zugunsten der Performance auf Plattenein/-ausgaben verzichtet, mit denen bei UTM-S die Sicherung von Benutzer- und Transaktionsdaten durchgeführt wird. Bei einer UTM-F-Anwendung sichert openUTM nur Benutzerkennworte und Änderungen in der Konfiguration, die mittels dynamischer Administration vorgenommen wurden. Diese Konfigurationsänderungen bleiben somit auch für den nächsten Anwendungslauf verfügbar. Änderungen an den Benutzerdaten dagegen werden in UTM-F-Anwendungen nicht gesichert. UTM-F-Anwendungen sind für Einsatzfälle geeignet, bei denen die Performance das wichtigste Kriterium ist, die Restart-Fähigkeit aber keine Rolle spielt. Dies trifft zu bei reinen Auskunftssystemen oder wenn alle Sicherungsfunktionen über das verwendete Datenbanksystem erbracht werden können.

APPLINAME=applinname

Name der UTM-Anwendung. *appliname* kann max. 8 Zeichen lang sein.

Pflichtoperand

Wenn, z.B. für die verteilte Verarbeitung über das LU6.1-Protokoll, mehrere Anwendungsnamen notwendig sind, so können Sie mit der BCAMAPPL-Anweisung weitere Anwendungsnamen für diese Anwendung vergeben. In APPLINAME= wird der primäre Anwendungsname festgelegt.

appliname muss im lokalen System eindeutig sein und darf nicht mit '$' beginnen.

B *BS2000/OSD:*

B Es gelten für *appliname* die Namenskonventionen für BCAM-Anwendungen.

B B B *appliname* darf nicht mit einer Ziffer beginnen, da BCAM dies nicht erlaubt und die Anwendung sonst nicht gestartet werden kann. Dabei ist zu beachten, dass KDCDEF etwaige Ziffern nicht abfängt.

X/W *Unix-, Windows-Systeme*:

X/W X/W *appliname* ist beim Verbindungsaufbau vom Terminal (Dialog-Terminal-Prozess) anzugeben.

X/W X/W X/W X/W Sollen über den mit APPLINAME= definierten Anwendungsnamen Verbindungen zu Partner-Anwendungen aufgebaut werden, dann müssen Sie für den Anwendungsnamen zusätzlich eine BCAMAPPL-Anweisung angeben, siehe hierzu die Beschreibung der BCAMAPPL-Anweisung auf Seite 289.

ASYNTASKS=(atask_number,service_number)

legt die Betriebsmittel fest, die maximal für die Bearbeitung von Asynchron-Aufträgen belegt werden dürfen.

atask_number

Maximale Anzahl der Prozesse (BS2000-Tasks bzw. Workprozesse unter Unix-/Windows-Systemen) der Anwendung, die gleichzeitig die Bearbeitung von Aufträgen an Asynchron-Transaktionscodes übernehmen. Mit diesem Operanden kann man verhindern, dass langlaufende Asynchron-Verarbeitungen den Dialog-Betrieb beeinträchtigen.

ASYNTASKS=0 bewirkt, dass keine Asynchron-TAC-Klassen generiert werden können.

Standard: 1
Minimalwert: 0
Maximalwert: TASKS -1

service_number

Maximale Anzahl der Asynchron-Vorgänge, die gleichzeitig offen sein dürfen.

Sie sollten *service_number* größer als *atask_number* wählen, wenn einer der beiden folgenden Fälle auftreten kann:

– Prozesswechsel während der Bearbeitung eines Asynchron-Vorgangs:
Besteht ein Asynchron-Vorgang aus mehreren Teilprogrammen und liegt der Transaktionscode eines Folgeprogramms (Folge-TAC nach PEND PA/PR oder PEND SP) in einer anderen TAC-Klasse als das aufrufende Teilprogramm oder ist die Prioritätensteuerung für TAC-Klassen generiert (Anweisung TAC-PRIORITIES), dann kann es bei der Bearbeitung zu einem Prozesswechsel kommen. Der Asynchron-Vorgang wird zunächst inaktiv und belegt keinen UTM-Prozess, bleibt aber offen.

– Asynchron-Vorgänge initiieren Dialoge mit LU6.1- oder OSI TP-Partner-Anwendungen:
Wird innerhalb eines Asynchron-Vorgangs ein Dialog mit einer Partner-Anwendung initiiert (mit APRO DM) und anschließend auf eine Antwort vom Partner gewartet (per PEND KP oder PEND RE), dann bleibt der Asynchron-Vorgang zwar offen bis die Antwort eintrifft (oder bis Timeout), belegt aber keinen UTM-Prozess.

Treten diese Fälle in der Anwendung auf und ist *service_number* zu klein gewählt, dann kann es vorkommen, dass die Asynchron-Verarbeitung vorübergehend blockiert wird, weil *service_number* inaktive Vorgänge existieren. Es können dann keine neuen Asynchron-Vorgänge gestartet werden, obwohl zu diesem Zeitpunkt kein UTM-Prozess einen Asynchron-Vorgang bearbeitet.

Standard: *atask_number*
Minimalwert: *atask_number*
Maximalwert: 32767

BLKSIZE= Größe einer UTM-Seite

Beachten Sie, dass abhängig von der BLKSIZE-Angabe jeder Benutzerspeicherbereich im Pagepool mindestens 2K bzw. 4K belegt.

Für UTM-Cluster-Anwendungen dürfen Sie nur BLKSIZE=4K angeben.

Standard bei stand-alone UTM-Anwendungen: 2K
Standard für UTM-Cluster-Anwendungen: 4K
mögliche Werte: 2K, 4K
Maximalwert: 4K

B B B

In BS2000/OSD müssen Sie BLKSIZE=4K angeben, wenn:
– die Dateien KDCFILE und USLOG auf NK4-Platten eingerichtet werden
– die KDCFILE als Hiperfile (High Performance File) genutzt werden soll.

X/W X/W X/W

Auf 32 Bit Unix-Systemen und auf Windows-Systemen müssen Sie BLKSIZE=4K angeben, wenn Sie mit mehr als 1976 Schlüsseln arbeiten wollen (MAX KEY=).

X X

Für UTM-Cluster-Anwendungen mit 64 Bit Unix-Systemen wird implizit eine UTM-Seitengröße von 4K eingestellt.

B B B B B

BRETRYNR=number

Anzahl der Versuche, die openUTM unternehmen soll, um eine Nachricht an das Transportsystem (BCAM) zu übergeben, wenn BCAM die Nachricht nicht sofort übernehmen kann. Wird diese Anzahl überschritten, wird die Verbindung zum Dialog-Partner abgebaut.

B B B B B B B B

Bei der Ausgabe von Asynchron-Nachrichten an einen Dialog-Partner mit PTYPE=APPLI (PTERM-Anweisung) hat BRETRYNR **keine** Bedeutung. Wird eine solche Nachricht vom Transportsystem wegen einer temporären Engpasssituation abgewiesen, dann gibt openUTM den Prozess zunächst frei, baut die Verbindung aber nicht ab. Nach einer Wartezeit von 3 Sekunden versucht openUTM erneut bis zu dreimal, die Nachricht an BCAM zu übergeben. Gelingt die Übergabe auch dann nicht, wird wiederum 3 Sekunden gewartet, bevor die nächsten 3 Versuche unternommen werden etc.

B B B

Standard: 10
Minimalwert: 1
Maximalwert: 32767 (theoretischer Wert)

X/W X/W X/W X/W X/W

CACHESHMKEY=number

Schlüssel für das Shared Memory Segment, in dem die anwendungsglobalen Puffer für die Dateizugriffe liegen. Die Schlüssel sind globale Parameter in Unix- bzw. Windows-Systemen. Es darf nur ein Schlüssel angegeben werden. *number* ist als Dezimalzahl anzugeben.

X/W

Pflichtoperand.

CACHESIZE=(number, paging, NORES oder RES)

B (NORES, RES nur in BS2000/OSD)

CACHESIZE= steuert Größe und Eigenschaften des Cache-Speichers (siehe auch openUTM-Handbuch „Konzepte und Funktionen“). Die Werte beeinflussen die Performance Ihrer UTM-Anwendung.

number Anzahl der UTM-Seiten des Cache-Speichers. Die Größe einer UTM-Seite wird im Operanden BLKSIZE= festgelegt. Über den Cache-Speicher werden alle Zugriffe auf den Pagepool abgewickelt, d.h. alle Ein- und Ausgaben von LSSBs, GSSBs, TLS, LPUT-Nachrichten und FPUT-Nachrichten, MPUT-Nachrichten an Clients, sowie einige UTM-Verwaltungsdaten. Das Schreiben auf KDCFILE erfolgt erst dann, wenn im Cache-Speicher kein Platz mehr ist oder wenn die Transaktion beendet wird.

KDCDEF rundet diese Zahl auf ein ganzzahliges Vielfaches von 32 auf.

Standardwert:
1024 (≙ 2 oder 4 MB, je nach Angabe bei BLKSIZE=)
Minimalwert:
32 (≙ 64 oder 128 KB, je nach Angabe bei BLKSIZE=; 1MB im oberen Adressraum des BS2000/OSD)
Maximalwert:
abhängig von Hardware und Betriebssystem, jedoch nicht größer als 16777184

B B B B B In BS2000/OSD liegt der Cache-Speicher in einem Common Memory Pool, dessen Größe immer ein Vielfaches von 1 MB beträgt. Das BS2000 rundet den mit CACHESIZE angegebenen Wert automatisch auf. Um keinen Adressraum zu verschenken, sollte CACHESIZE in Vielfachen von 1 MB angefordert werden.

paging gibt an, wieviel Prozent der Seiten des Cache-Speichers bei Engpasssituationen auf einmal auf die KDCFILE geschrieben werden sollen, um den Cache-Speicher zu entlasten. Es werden jedoch mindestens 8 Seiten ausgelagert. Dieser Wert kann mit dem Administrationskommando KDCAPPL CACHE=*%_utm_pages* geändert werden.

Standardwert: 70 (%)
Minimalwert: 0, d.h. es werden 8 Seiten ausgelagert
Maximalwert: 100 (%)

B NORES Der Cache-Speicher soll pageable (nicht resident) angelegt werden.

B Standard: NORES

RES Der Cache-Speicher soll resident angelegt werden.
Im Produktionsbetrieb sollte zur Verbesserung der Performance mit einem residenten Cache-Speicher gearbeitet werden, weil der Cache-Paging-Algorithmus danach ausgerichtet ist.

Der Verbrauch residenter Seiten durch das Einrichten eines residenten Cache-Speichers kann nicht mit dem COREBIAS-Operanden des BS2000-Kommandos BIAS kontrolliert werden.

CARDLTH=laenge

Länge der Ausweisinformation in Bytes. Wenn der Ausweisleser zur Ergänzung der Berechtigungs-Prüfung verwendet wird, speichert openUTM die Ausweisinformation in der Länge ab, die sich aus dem Maximum dieser Länge und dem bei MAX PRINCIPAL-LTH generierten Wert ergibt.
Wenn die Information auf dem Ausweis länger ist, wird sie auf diese Länge verkürzt abgespeichert.
Mit dem KDCS-Aufruf INFO (KCOM=CD) kann der Teilprogrammlauf diese Information lesen.

CARDLTH muss so groß sein, dass für alle USER Anweisungen mit USER ..., CARD = (*pos, string*) gilt:

pos + Laenge (string) -1 ≤ CARDLTH.

Standard: 0
Maximalwert: 255

Bei einem Wert > 255 nimmt KDCDEF ohne Warnung 255 an.

CATID=(catalog_A,catalog_B)
legt fest, welchen Katalogkennungen Ihre KDCFILE zugeordnet wird.

Wenn Sie mit CATID arbeiten, geben Sie bei KDCFILE=*filebase* (siehe Seite 383) den Basisnamen ohne die CATID an.

Solange Sie die KDCFILE einfach führen, geben Sie mit *catalog_A* die CATID an, der die KDCFILE zugeordnet werden soll. *catalog_B* wird in diesem Fall nicht angegeben.

Bei doppelter Dateiführung der KDCFILE (siehe Abschnitt „Die KDCFILE“ auf Seite 44) können Sie Dateien mit dem Suffix A der CATID *catalog_A* zuordnen und Dateien mit dem Suffix B der CATID *catalog_B*. Geben Sie bei doppelter Dateiführung nur *catalog_A* an, werden jeweils beide Dateien dieser CATID zugeordnet.

CLRCH= definiert ein Zeichen, mit dem der KB-Programmbereich und der SPAB am Ende eines Dialog-Schrittes überschrieben werden. Mögliche Angaben sind:

c
C'c'
X'xx'

Dabei ist *c* ein alphanumerisches und *x* ein hexadezimales Zeichen.

Standard: Die Bereiche KB und SPAB werden nicht überschrieben.

CONN-USERS=number

Auslastung der Anwendung steuern
CONN-USERS= legt die maximale Anzahl der Benutzer fest, die gleichzeitig mit der Anwendung arbeiten dürfen. Bei einer Anwendung, für die keine Benutzerkennungen generiert sind, kann mit CONN-USERS= die maximale Anzahl der Clients festgelegt werden, die sich gleichzeitig über LTERM-Partner an die Anwendung anschließen dürfen.

- CONN-USERS < Anzahl Benutzer/Clients:
  verhindert, dass alle Benutzer/Clients gleichzeitig mit der Anwendung arbeiten.
- CONN-USERS=0:
  Die Anzahl der gleichzeitig aktiven Benutzer/Clients wird nicht beschränkt.
- CONN-USERS > Anzahl Benutzer/Clients:
  keine Steuerung der Auslastung, CONN-USERS= hat keine Wirkung.

Benutzerkennungen und Clients, die mit Administrationsberechtigung generiert wurden, können sich auch dann noch an die UTM-Anwendung anmelden, wenn die maximale Anzahl der gleichzeitig arbeitenden Benutzerkennungen bereits erreicht wurde.

B

Standardwert in BS2000/OSD: 0 (d.h. keine Beschränkung)
Minimalwert: 0
Maximalwert: 500000

X/W
X/W
X/W

CONN-USERS ist in Unix- und Windows-Systemen Pflichtoperand. Bitte beachten Sie, dass *number* nicht höher gesetzt werden darf als die Anzahl der erworbenen Concurrent-User-Lizenzen.

CONRTIME=time

(**con**nection **r**equest **time**) Zeit in Minuten, nach der openUTM bei Ausfall der logischen Verbindung versucht, die Verbindung zu den folgenden Partnern zyklisch wieder aufzubauen:

– TS-Anwendungen (PTYPE=APPLI oder PTYPE=SOCKET), die mit automatischem Verbindungsaufbau durch openUTM (PTERM ...,CONNECT=YES) generiert sind, wenn die Verbindung nicht durch die Administration oder wegen Ablauf des Timers IDLETIME (siehe PTERM-Anweisung auf Seite 440) abgebaut wurde.

– OSI TP- oder LU6.1-Partner-Anwendungen, die mit automatischem Verbindungsaufbau generiert sind, wenn die Verbindung nicht durch die Administration oder wegen Ablauf des Timers IDLETIME abgebaut wurde.

– OSI TP-Partner, an die Asynchron-Nachrichten gesendet wurden und zu denen zum Erzeugungszeitpunkt der Nachricht keine Verbindung bestand.

B – Drucker, zu denen openUTM eine Verbindung aufbaut, sobald die Anzahl der Druckaufträge für diesen Drucker den generierten Schwellwert überschreitet (LTERM ...,PLEV>0). Die Anzahl der Druckaufträge muss bei Verbindungsabbruch größer oder gleich dem Schwellwert sein, damit openUTM versucht, die Verbindung wieder aufzubauen. Bei CONRTIME≠0 versucht openUTM die Verbindung zum Drucker auch dann aufzubauen, wenn diese vorher explizit durch die Administration abgebaut wurde.

B – Drucker, zu denen openUTM automatisch eine Verbindung aufbaut (PTERM ...,CONNECT=YES) und wenn die Verbindung nicht durch die Administration abgebaut wurde.

B – Nachrichtenverteiler (MUX), zu denen openUTM beim Start automatisch eine Verbindung aufbaut, wenn diese Verbindung nicht zuvor durch die Administration abgebaut wurde.

Kommt beim Start der Anwendung oder per Administrationskommando KDCPTERM bzw. KDCLPAP keine Verbindung zu diesen Partnern zustande, so versucht auch in diesem Fall openUTM im Abstand von CONRTIME= die Verbindung wiederaufzubauen.

Bei CONRTIME=0 macht openUTM keinen Versuch die Verbindung aufzubauen. *Ausnahme*: Bei Asynchron-Nachrichten an OSI TP-Partner wird dann eine Wartezeit von 10 Minuten gesetzt.

Standard: 10 Min.
Maximalwert: 32767 Min.

DEAD-LETTER-Q-ALARM

steuert die Überwachung der Anzahl von Nachrichten in der Dead Letter Queue.

Bei jedem Erreichen des Schwellwertes *number* wird die Meldung K134 ausgegeben. Für diese Meldung kann das Ziel MSGTAC definiert werden, um eine automatische Behandlung der Dead Letter Queue durchzuführen.

Standard: 0, die Überwachung ist ausgeschaltet.
Minimalwert: 0
Maximalwert: 65535

DESTADM=destination

gibt den Empfänger an, an den openUTM die Ergebnisse von Administrationsaufrufen schickt, die asynchron verarbeitet wurden. Für *destination* kann angegeben werden:

- ein LTERM-Partner
  Ausnahme: UPIC-LTERM-Partner sind nicht zulässig!
- der TAC eines Asynchron-Programms
- eine TAC-Queue (Type=Q)

Standard: Leerzeichen, d.h. kein Ziel, die Ergebnisse gehen verloren.

DPUTLIMIT1=(day,hour,minute,second)

definiert die spätestmögliche Ausführungszeit eines Auftrags bei relativer oder absoluter Zeitangabe:

Ausführungszeitpunkt < DPUT-Aufrufzeitpunkt + DPUTLIMIT1

Für die Zeitangabe in DPUTLIMIT1 gilt:

day	Maximalwert: 364 Minimalwert: 0
hour	Maximalwert: 23 Minimalwert: 0
minute	Maximalwert: 59 Minimalwert: 0
second	Maximalwert: 59 Minimalwert: 0

Standardwert: DPUTLIMIT1 (360, 0, 0, 0) = 360 Tage
Standardwert: DPUTLIMIT2 (1, 0, 0, 0) = 1 Tag
Minimalwert: (0, 0, 0, 0)
Maximalwert: (364, 23, 59, 59)

Für die Operanden DPUTLIMIT1 und DPUTLIMIT2 muss gelten:

DPUTLIMIT1 + DPUTLIMIT2 ≤ (364, 23, 59, 59) < 365 Tage

D.h. geben Sie für DPUTLIMIT1 (364, 23, 59, 59) an, so müssen Sie DPUTLIMIT2=(0, 0, 0, 0) angeben.

DPUTLIMIT2=(day,hour,minute,second)

Die Zeitangabe beim DPUT-Aufruf enthält keine Jahreszahl. Außerdem kann der gewünschte Ausführungszeitpunkt bereits vorüber sein, wenn sich der DPUT-Aufruf verzögert hat.

Daher muss entschieden werden, ob der Ausführungszeitpunkt eines Auftrags mit absoluter Zeitangabe dem vergangenen, laufenden oder nächsten Jahr zugerechnet werden soll.

Da gilt, dass DPUTLIMIT1 + DPUTLIMIT2 < 1 Jahr sein müssen, liegt höchstens eine dieser drei Alternativen im erlaubten offenen Zeitintervall (Aufrufzeitpunkt - DPUTLIMIT2, Aufrufzeitpunkt + DPUTLIMIT1):

- Liegt die einzig erlaubte Alternative vor dem Aufrufzeitpunkt, wird der DPUT als FPUT behandelt und schnellstmöglich ausgeführt.
- Liegt die einzig erlaubte Alternative nach dem Aufrufzeitpunkt, so wird der DPUT gesichert und erst zu dieser Zeit in einen FPUT umgewandelt und ausgeführt.
- Liegt keine der drei Alternativen im erlaubten Zeitintervall, so wird der DPUT abgewiesen.

DPUTLIMIT2 ermöglicht also bei einem zeitgesteuerten Auftrag mit absoluter Zeit-Angabe die Rückdatierung des angegebenen Ausführungszeitpunkts in die Vergangenheit. Bei relativer Zeitangabe gibt es keine Rückdatierung.

DPUTLIMIT1 begrenzt die Vordatierung von Aufträgen mit relativer oder absoluter Zeitangabe in die Zukunft.

Beispiel 1

```
DPUTLIMIT1 = (300,0,0,0)
DPUTLIMIT2 = (010,0,0,0)
```

Der DPUT-Aufrufzeitpunkt ist (005,0,0,0). Vergangenes und aktuelles Jahr sind keine Schaltjahre.

- DPUTs mit relativer Zeit (000,0,0,0) bis (299,23,59,59) werden angenommen.
- DPUTs mit Absolutzeit (001,0,0,0) bis (005,0,0,0) und (360,0,0,1) bis (365,23,59,59) werden als FPUT behandelt.
- DPUTs mit Absolutzeit (005,0,0,1) bis (304,23,59,59) werden als DPUT behandelt.
- DPUTs mit Absolutzeit (305,0,0,0) bis (360,0,0,1) werden abgewiesen.

```
-|--------------|------------ ... -------------|----------|
360            5                             305        360
 FFFFFFFFFFFFFFFFDDDDDDDDDDDDD ... DDDDDDDDDDDDDD
```

Beispiel 2

DPUTLIMIT1 und DPUTLIMIT2 sind genauso definiert wie in *Beispiel 1*, der DPUT-Aufrufzeitpunkt ist aber (360,0,0,0).

- DPUTs mit relativer Zeit (000,0,0,0) bis (299,23,59,59) werden angenommen.
- DPUTs mit Absolutzeit (350,0,0,1) bis (360,0,0,0) werden als FPUT behandelt.
- DPUTs mit Absolutzeit (001,0,0,0) bis (294,23,59,59) und (360,0,0,1) bis (365,23,59,59) werden als DPUT behandelt.
- DPUTs mit Absolutzeit (295,0,0,0) bis (350,0,0,0) werden abgewiesen.

```
-|--------------|------------- ... ----------|-------------|
 350           360                          295           350
  FFFFFFFFDDDDDDDDDDDDDDDDDDDDD ... DDDDDDDDDD
```

Standardwerte siehe Operand DPUTLIMIT1

GSSBS=number

Maximale Anzahl von GSSBs (globale Sekundärspeicherbereiche), die gleichzeitig in der Anwendung existieren können.

> **i** In UTM-Cluster-Anwendungen, die mit CLUSTER GLOBAL-UTM-DATA=NO (Standard) generiert sind, muss GSSBS=0 angegeben werden.

Standard: 32
Minimalwert: 0
Maximalwert: 30000

HOSTNAME=name

darf nur in stand-alone-Anwendungen angegeben werden.

In UTM-Cluster-Anwendungen können Sie einen virtuellen Rechnernamen im Parameter VIRTUAL-HOST der CLUSTER-NODE-Anweisung angeben.

B *BS2000/OSD:*
B Name des virtuellen Hosts, auf dem (aus BCAM-Sicht) die UTM-Anwendung läuft. Dieser virtuelle Host muss auch in BCAM generiert sein.
B Standard: 8 Leerzeichen, d.h. die Anwendung läuft unter dem Namen des realen Rechners.

X/W *Unix-, Windows-Systeme:*
X/W Name des Hosts, der als Absenderadresse angegeben wird, wenn eine Verbindung von der UTM-Anwendung aus aufgebaut wird. HOSTNAME= wird in Cluster-Systemen benötigt, die beim Verbindungsaufbau als Absenderadresse die übernehmbare („relocatable“) IP-Adresse und nicht die stationäre IP-Adresse verwenden.

X/W Standard: 8 Leerzeichen, d.h. als Absenderadresse wird der Standard-
X/W Rechnername des Transportsystems verwendet.

X/W IPCSHMKEY=number
X/W Schlüssel für das Shared Memory Segment, das zur Kommunikation zwi-
X/W schen Workprozessen einerseits und den Dialog-Terminal- bzw. Drucker-
X/W Prozessen sowie dem Timerprozess (externe Prozesse einer Anwendung)
X/W andererseits dient. Die Schlüssel sind globale Parameter in Unix- bzw. Win-
X/W dows-Systemen. Es darf nur ein Schlüssel angegeben werden. *number* ist
X/W als Dezimalzahl anzugeben.

X/W Pflichtoperand.

X/W IPCTRACE=number
X/W Im Testmodus (Starten mit TESTMODE=ON; openUTM-Handbuch „Ein-
X/W satz von openUTM-Anwendungen unter Unix- und Windows-Systemen“)
X/W schreibt openUTM Einträge in den Trace-Bereich des IPC (Shared Memory
X/W Segments für die Kommunikation zwischen Prozessen). Diese Einträge
X/W enthalten interne Informationen für Diagnosezwecke. Ein Eintrag belegt 32
X/W Byte. Mit IPCTRACE wird die Anzahl von Einträgen festgelegt. Wird diese
X/W Zahl überschritten, so überschreibt openUTM bereits vorhandene Einträge
X/W vom ältesten an.

X/W Standard: 1060
X/W Minimalwert: 1
X/W Maximalwert: 32500

X/W Eine Angabe < 1 oder > 32500 wird von KDCDEF ohne Meldung durch den
X/W Minimal- bzw. Maximalwert ersetzt.

X/W KAASHMKEY=number

X/W X/W X/W X/W X/W Schlüssel für das Shared Memory Segment, in dem die anwendungsglobalen Daten abgelegt werden. Die Schlüssel sind globale Parameter in Unix- bzw. Windows-Systemen. Es darf nur ein Schlüssel angegeben werden. *number* ist als Dezimalzahl anzugeben.

X/W Pflichtoperand.

KB=length legt die Länge des Kommunikationsbereiches (KB) in Byte fest. KB-Kopf und KB-Rückgabebereich werden bei der Länge nicht berücksichtigt.

Standard: 512
Minimalwert: 0
Maximalwert: 32767

KDCFILE=

filebase Basisname der KDCFILE, der Benutzer-Protokolldatei und der System-Protokolldatei SYSLOG.
filebase muss auch beim Start der Anwendung im Startparameter FILEBASE=*filebase* angegeben werden (siehe openUTM-Handbuch „Einsatz von openUTM-Anwendungen“).

Pflichtoperand.

B B B B B *BS2000/OSD:*
Wenn Sie den Parameter CATID= nutzen, um Ihrer KDCFILE Katalogkennungen zuzuordnen, muss der Basisname ohne CATID angegeben werden (zu Aufbau und Länge der Namen siehe Abschnitt „BS2000/OSD:“ auf Seite 45).

X/W X/W X/W X/W X/W X/W X/W X/W X/W *Unix-, Windows-Systeme:*
filebase ist das Dateiverzeichnis, das die KDCFILE und alle Dateien der Anwendung enthält. Das Dateiverzeichnis muss **vor** dem KDCDEF-Lauf eingerichtet werden.
filebase kann vollqualifiziert oder teilqualifiziert angegeben werden und darf für stand-alone Anwendungen maximal 29 Zeichen lang sein, unabhängig davon, ob der Name voll- oder teilqualifiziert angegeben wird.
Für UTM-Cluster-Anwendungen darf *filebase* maximal 27 Zeichen lang sein.

SINGLE Die KDCFILE soll einfach geführt werden.

Wird die KDCFILE aufgespalten (siehe Abschnitt „KDCFILE aufspalten“ auf Seite 56), so werden alle KDCFILE-Dateien einfach geführt.

Standard: SINGLE

DOUBLE Die KDCFILE soll aus Sicherheitsgründen doppelt geführt werden.

Wird die KDCFILE aufgespalten (siehe Abschnitt „KDCFILE aufspalten“ auf Seite 56), so werden alle KDCFILE-Dateien doppelt geführt.

KEYVALUE=number

Wert des größten Keycodes der Anwendung und damit auch Wert des größten entsprechenden Lockcodes, der für einen Transaktionscode oder ein Terminal als Zugriffschutz vergeben werden kann.
Mit KEYVALUE=*number* ist auch die maximale Anzahl der Keycodes pro Keyset festgelegt. openUTM verwendet diese Angabe zur Optimierung der Keyset-Tabellen.
Es lassen sich maximal 4000 Lock- und Keycodes definieren. Es können nur numerische Lockcodes definiert werden.

Standardwert: 32
Minimalwert: 1

B	
X/W	Maximalwert (BS2000/OSD): 4000
X/W	Maximalwert (32 Bit Unix-, Windows-Systeme):
X/W	4000 bei MAX ...,BLKSIZE=4K
X/W	1976 bei MAX ...,BLKSIZE=2K
X	Maximalwert (64-Bit-Unix-Systeme):
X	4000 für stand-alone UTM-Anwendungen
X	3900 für UTM-Cluster-Anwendungen

Bei einem Wert < 1 setzt KDCDEF ohne Meldung KEYVALUE=1.

LEADING-SPACES=

gibt an, wie führende Leerzeichen in einer Nachricht von einem Terminal oder von einer TS-Anwendung (PTERM ... PTYPE=APPLI oder SOCKET) behandelt werden.

YES Führende Leerzeichen in einer Nachricht werden beim Aufruf eines Teilprogramms an das Teilprogramm weitergereicht.
Es wird ein Leerzeichen zwischen TAC und Nachricht als Trennzeichen entfernt, wenn der TAC-Name < 8 Zeichen ist.

Beispiel

Bei der Nachricht `TACNAME␣␣Message` wird dem TAC TACNAME bei FGET/MGET folgende Nachricht an das Teilprogramm zugestellt:

```
␣Message
```

NO Führende Leerzeichen werden ausgeblendet.

Standard: NO

B LOCALE=(lang_id,terr_id,ccsname)

B definiert die Standard-Sprachumgebung der UTM-Anwendung (siehe auch
B Abschnitt „UTM-Meldungen“ auf Seite 183).

B Das hier generierte Locale wird den Benutzerkennungen und den Clients,
B die sich über LTERM-Partner oder LTERM-Pool anschließen, als Standard-
B wert für die Sprachumgebung zugeordnet. Die Standardeinstellung ist wirk-
B sam, solange für diese Objekte in den entsprechenden USER-, LTERM-
B oder TPOOL-Anweisungen kein eigenes Locale definiert wird.

B Das Meldungsmodul, dessen Sprach- und Territorialkennzeichen in den
B Anweisungen MESSAGE ...,LOCALE= und MAX ...,LOCALE= überein-
B stimmen, wird das Anwendungs-Meldungsmodul. Meldungen an die Mel-
B dungsziele SYSOUT, SYSLST und CONSOLE sendet openUTM aus die-
B sem Anwendungs-Meldungsmodul. Außerdem bestimmen die Angaben im
B Anwendungs-Meldungsmodul die Meldungsziele für eine Meldung.

B lang_id Max. zwei Zeichen langes Sprachkennzeichen für die UTM-Anwendung.
B Das Sprachkennzeichen ist frei wählbar.

B Standard: Leerzeichen

B terr_id Max. zwei Zeichen langes Territorialkennzeichen. Das Territorialkenn-
B zeichen ist frei wählbar.

B Standard: Leerzeichen

B ccsname (**c**oded **c**haracter **s**et **name**)
B Für *ccsname* ist der max. 8 Zeichen lange Name eines erweiterten Zeichen-
B satzes (CCS-Name) anzugeben. Der angegebene CCS-Name muss zu ei-
B nem im BS2000-System definierten EBCDIC-Zeichensatz gehören (siehe
B auch Benutzerhandbuch zu XHCS). Zum Zeitpunkt der Generierung kann
B openUTM diese Bedingung nicht überprüfen; KDCDEF akzeptiert deshalb
B auch CCS-Namen, denen kein Zeichensatz zugeordnet ist.

B Standard: Leerzeichen, d.h. 7-Bit-Betrieb

B LOGACKWAIT=time

B B Zeit in Sekunden, die openUTM maximal auf eine Quittung von Ausgabegeräten warten soll. Diese Quittung ist

B B B B
- bei einem Drucker die logische Abdruckquittung vom Drucker,
- bei einem RSO-Drucker die Quittung von RSO,
- bei einem FPUT-Aufruf an eine andere Anwendung die Transportquittung.

B B Trifft in diesem Zeitraum die Quittung nicht ein, z.B. wegen Papierende bei Druckern, baut openUTM die logische Verbindung zu dem Gerät ab.

B B B Standard: 600
Minimalwert: 10
Maximalwert: 32767

LPUTBUF=number

Größe des LPUT-Puffers in UTM-Seiten. Im LPUT-Puffer der KDCFILE werden die LPUT-Daten zwischengespeichert. Diese LPUT-Daten werden erst dann in die Benutzer-Protokolldatei kopiert, wenn *number* überschritten wird. Die Benutzer-Protokolldatei USLOG ist nur während dieses Kopier-Vorgangs geöffnet.

Standard: 1
Minimalwert: 1
Maximalwert: 1000
Einen Wert > 1000 ersetzt KDCDEF ohne Meldung durch 1000.

VORSICHT!
Dieser Operand sollte unbedingt > 1 eingestellt werden, falls in der Anwendung LPUT-Aufrufe vorkommen. Sonst wird der Kopier-Vorgang zu oft gestartet. Das ist jeweils mit einem Öffnen und Schließen der Benutzer-Protokolldateien verbunden.
Die Angabe bei LPUTBUF muss so gewählt werden, dass auch der längste LPUT-Satz in den Puffer passt. Es muss gelten:

LPUTBUF ∗ Größe UTM-Seite ≥ LPUTLTH + Länge KB-Kopf (84 Byte)

LPUTLTH=length

Maximale Länge der Benutzerdaten in LPUT-Sätzen in Byte (ohne KB-Kopf).

Die maximale Länge eines Satzes in der Benutzer-Protokolldatei ergibt sich damit aus (siehe auch openUTM-Handbuch „Anwendungen programmieren mit KDCS“, Benutzer-Protokolldatei):

length + 84 Byte (KB-Kopf) + 12 Byte (Längenfelder)

Standard: 1948
Minimalwert: 0

B
B
X/W
Maximalwert (BS2000/OSD): 32652, unabhängig vom Speichermedium der Benutzerprotokolldatei
Maximalwert (Unix- und Windows-Systeme): 32668

B
BS2000/OSD:

B
B
B
B
B
B
Aus *length* bestimmt openUTM darüber hinaus die Blocklänge der Benutzer-Protokolldatei. Dazu ermittelt openUTM zu dem Wert (l*ength* + 100 Byte) das nächstgrößere Vielfache von 2 KByte. Dieses Vielfache nimmt openUTM als Blocklänge für die Benutzer-Protokolldatei. Die 100 Byte setzten sich zusammen aus 84 Byte für KB-Kopf + 12 Byte für Satzlängenfelder + 4 Byte für die Blocklängenfelder.

B
B
Soll die Benutzer-Protokolldatei USLOG auf einer Non-Key-Platte (NK2, NK4) eingerichtet werden, dann müssen Sie *length* so wählen, dass:

B
B
length + 100 Byte
+ 16 Byte Block-spezifische DVS-interne Verwaltungsinformation

B
B
gleich einem Vielfachen von 2 KByte (auf NK2-Platten) oder 4 KByte (auf NK4-Platten) ist. Damit kann der Plattenplatz optimal genutzt werden.

B
B
B
Die 16 Byte Block-spezifische DVS-Verwaltungsinformation stehen somit für Benutzerdaten nicht zur Verfügung. Nähere Information dazu findet sich im BS2000-Handbuch „Einführung in das DVS“.

LSSBS=number

Maximale Anzahl von LSSBs (lokale sekundäre Speicherbereiche), die in einem Vorgang erzeugt werden können.

Standard: 8
Minimalwert: 0
Maximalwert: 1600

B
B
B

MP-WAIT=number

gibt an, wieviele Sekunden openUTM maximal auf den Anschluss eines Prozesses an einen Common Memory Pool wartet.

B
B
B

Standardwert: 180
Minimalwert: 1
Maximalwert: 32000

B
B
B
B

VORSICHT!
Der Standardwert von 180 Sekunden sollte nur im Ausnahmefall geändert werden, wenn sich beispielsweise ein Prozess mit K078 ENQAR und einem Userdump mit dem Returncode KDCSST01 beendet.

NB=length

legt die maximale Länge eines Arbeitsbereiches fest für:
- logische Ein- und Ausgaben von und zu Terminals und Transportsystem-Anwendungen vom Typ APPLI
- asynchrone Ausgabe-Nachrichten an Drucker und Transportsystem-Anwendungen vom Typ SOCKET

Anzugeben ist die Länge des größten Nachrichtenbereiches der Teilprogramme in Byte.
Der bei NB angegebene Wert darf in BS2000/OSD den Wert von TRMSGLTH, in Unix- und Windows-Systemen den Wert TRMSGLTH - 24 nicht überschreiten (siehe MAX ...,TRMSGLTH=).

Standard: 2048
Minimalwert: 2048

B
X/W

Maximalwert (BS2000/OSD): 32700
Maximalwert (Unix-,Windows-Systeme): 32676

B
B
B
B
B

BS2000/OSD:
Wenn RSO-Drucker definiert werden, muss die Größe des RSO-Puffers (REMOTE-BUFFER-SIZE in der SPOOL-Parameterdatei) größer gleich *length* sein. Siehe auch Abschnitt „RSO-Drucker generieren (BS2000/OSD)“ auf Seite 169.

X/W
X/W
X/W

NET-ACCESS=

gibt an, ob die Netzanbindung der Anwendung multi-threaded oder single-threaded erfolgt.

X/W
X/W

Die Angabe in NET-ACCESS hat keine Auswirkung auf Verbindungen über die Socket-Schnittstelle (native TCP/IP generiert mit T-PROT=SOCKET).

X/W X/W MULTI-THREADED

Es werden mehrere Netzverbindungen in einem Netzprozess verwaltet.

X/W X/W X/W X/W X/W X/W X/W X/W X/W Die Zuordnung von Netzverbindung zu Netzprozess erfolgt über Listener-IDs, die Sie den Anwendungsnamen (BCAMAPPL-Anweisung, siehe Seite 289) und Zugriffspunkten (ACCESS-POINT-Anweisung, siehe Seite 275) Ihrer Anwendung zuordnen. Alle Verbindungen mit derselben Listener-ID werden von demselben Netzprozess verwaltet. Wenn Sie NET-ACCESS=MULTI-THREADED angeben, sollten Sie auch Listener-IDs generieren, da sonst alle Zugriffspunkte und Anwendungsnamen implizit die Listener-ID 0 erhalten und alle von Verbindungen demselben Netzprozess verwaltet werden.

X/W Standard: MULTI-THREADED

X/W X/W SINGLE-THREADED

Jede Netzverbindung wird in einem eigenen Netzprozess verwaltet.

W W W W Diese Art der Netzanbindung wird unter Windows-Systemen nicht unterstützt. Geben Sie in einer UTM-Anwendung, die unter Windows-Systemen ablaufen soll, SINGLE-THREADED an, dann ersetzt KDCDEF den Wert ohne Meldung durch MULTI-THREADED.

NRCONV=number

(**n**umbe**r** of **conv**ersations)
Maximalzahl der Vorgänge, die ein Benutzer gleichzeitig kellern darf.
NRCONV=0 bedeutet, dass kein Vorgang gekellert werden kann.

Es gilt folgender Grenzwert:

Anzahl der Benutzerkennungen
+ Anzahl der Vorgänge, die maximal gekellert werden dürfen
(Anzahl der Vorgänge=*number* ∗ Anzahl der Benutzerkennungen)
≤ 500000

Wird der Grenzwert 500000 überschritten (durch die Angaben bei NRCONV, in der RESERVE-Anweisung, siehe Seite 464, und durch die Anzahl der USER-Anweisungen, siehe Seite 538), dann legt openUTM automatisch weniger Einträge für die Vorgangskellerung an. Es können dann nicht alle Benutzer gleichzeitig *number* viele Vorgänge kellern.

Standard: 0
Minimalwert: 0
Maximalwert: 15

X/W X/W X/W X/W OSISHMKEY=number

Schlüssel für das Shared Memory Segment, das bei der Kommunikation über OSI TP von OSS verwendet wird. Für *number* ist der Schlüssel als Dezimalzahl anzugeben.

X/W X/W OSISHMKEY ist Pflichtoperand, falls die Anwendung über OSI TP kommuniziert.

OSI-SCRATCH-AREA=value

Größe in KB eines UTM-internen Arbeitsbereichs zur dynamischen Ablage von Daten, wenn mit dem OSI TP-Protokoll gearbeitet wird.

Standard: 256
Minimalwert: 128
Maximalwert: 32767

B B In BS2000/OSD wird der Arbeitsbereich während des Anwendungslaufs gegebenenfalls automatisch vergrößert.

X/W X/W X/W X/W X/W In Unix- und Windows-Systemen ist die Größe des internen Arbeitsbereichs während des Anwendungslauf nicht zu ändern. Es wird empfohlen, mit dem Standardwert zu arbeiten. Wenn sich trotzdem beim Betrieb der interne Arbeitsbereich als nicht ausreichend erweist, muss die Generierung mit einem höheren Wert für OSI-SCRATCH-AREA wiederholt werden.

PGPOOL=(number,warnlevel1,warnlevel2)

legt die Größe des Pagepools in Anzahl UTM-Seiten und die Warnstufen für die Pagepool-Belegung fest.

number Anzahl der UTM-Seiten, die für den Pagepool in der KDCFILE verwendet werden sollen (siehe Abschnitt „Pagepool“ auf Seite 48). Die Größe einer UTM-Seite wird im Operanden BLKSIZE= generiert.

Standard: 100
Minimalwert: 20
Maximalwert: 16777215 - (2 ∗ Anzahl PGPOOLFS)

Bei einem Wert < 20 setzt KDCDEF ohne Meldung PGPOOL=20.

X/W X/W In Unix- und Windows-Systemen nimmt PGPOOL immer gerade Werte an. Werden ungerade Werte angegeben, zieht openUTM von der Angabe 1 ab.

warnlevel1

Numerischer Wert (Prozentwert), der angibt, bei welcher Belegung des Pagepool die erste Warnung (Meldung K041) ausgegeben wird.

Standard: 80
Minimalwert: 1
Maximalwert: 99

warnlevel2

Numerischer Wert (Prozentwert), der angibt, bei welcher Belegung des Pagepool die zweite Warnung ausgegeben werden soll. Nach Überschreiten von *warnlevel2* werden Asynchron-Aufträge abgewiesen. In diesem Fall erhält der Benutzer eine K-Meldung und einem Teilprogramm wird ein entsprechender Returncode übermittelt.

Standard: 95
Minimalwert: *warnlevel1* + 1
Maximalwert: 100

PGPOOLFS=number

Anzahl der Dateien, auf die der Pagepool aufgeteilt werden soll. Bei PGPOOLFS=0 liegt der Pagepool in der Hauptdatei (unter BS2000/OSD in der Datei filebase.KDCA, unter Unix- und Windows-Systemen in der Datei KDCA im Dateiverzeichnis *filebase)*. Zweite Exemplare bei doppelter Dateiführung (MAX ...,KDCFILE=(...,DOUBLE)) werden bei *number* nicht mitgezählt.
Die Dateinamen vergibt KDCDEF.

Standard: 0, d.h. der Pagepool liegt in der Hauptdatei

Beachten Sie, dass BS2000-Dateien nicht größer als 32 GByte sein können.

B
X/W

Maximalwert (BS2000/OSD): 99 (bzw. PGPOOL=*number* / 2)
Maximalwert (Unix-, Windows-Systeme): 10

B
B
B
X/W
X/W
X
X
X

Minimalwert:
Im BS2000/OSD darf eine einzelne UTM-Datei nicht größer als 32 GBytes werden.
Auf Unix-Systemen im 32 Bit-Modus und Windows werden Dateien bis 2 GByte Größe unterstützt.
Auf Unix-Systemen im 64 Bit-Modus kann openUTM auch größere Dateien im Rahmen der vom Betriebssystem und Filesystem möglichen Grenzen verwenden.

Daher ergibt sich folgender Minimalwert in Abhängigkeit von der Größe einer UTM-Seite:
2, falls BLKSIZE = 4K und PGPOOL *number* ≥ 8388608
0: Der Pagepool liegt in der Hauptdatei.

PGWTTIME=time

Zeit in Sekunden, die ein Teilprogramm nach einem blockierenden Aufruf (z.B. PGWT-Aufruf) maximal auf das Eintreffen von Nachrichten warten darf. Während dieser Wartezeit bleibt ein Prozess der UTM-Anwendung exklusiv für dieses Teilprogramm reserviert.

Standard: ≙ *time* in TERMWAIT=*time*
Minimalwert: 60
Maximalwert: 32767

B PRINCIPAL-LTH=length

B Maximale Länge eines Kerberos-Principals in Byte.
B Der Parameter ist nur von Bedeutung, wenn mindestens ein Benutzer mit
B USER ..., PRINCIPAL= oder mindestens ein LTERM oder TPOOL mit
B KERBEROS-DIALOG=YES generiert ist. Die Länge des bei USER ...
B PRINCIPAL= angegebenen Wertes darf nicht größer sein als der bei MAX
B PRINCIPAL-LTH= generierte Wert.

B Wenn ein Kerberos-Dialog mit einem Client durchgeführt wird, speichert
B openUTM die Kerberos-Information in der Länge ab, die sich aus dem Ma-
B ximum dieser Länge und der bei MAX CARDLTH generierten Länge ergibt.
B Wenn die Kerberos-Information länger ist, wird sie auf diese Länge verkürzt
B abgespeichert.
B Mit dem KDCS-Aufruf INFO (KCOM=CD) kann der Teilprogrammlauf diese
B Information lesen, falls sich nach dem Kerberos-Dialog mit einem Client am
B selben Client kein Benutzer mit Ausweiskarte angemeldet hat, da in diesem
B Fall die Kerberos-Information mit der Ausweis-Information überschrieben
B worden ist.

B Standard: 0
B Minimalwert: 0
B Maximalwert: 100

QTIME = (qtime1, qtime2)

Legt die Zeit fest, die ein Vorgang maximal auf das Eintreffen von Nachrichten für Service-gesteuerte Queues warten darf. Die Zeiten gelten für USER-Queues, TAC-Queues und Temporäre Queues.

Für das Warten in Dialog- bzw. Asynchron-Vorgängen kann jeweils ein eigener Maximalwert definiert werden.

Wird in einem Teilprogrammlauf eine größere Wartezeit angegeben als die in QTIME= generierte, dann setzt openUTM die Wartezeit auf den generierten Wert zurück.

qtime1 maximale Wartezeit in Dialog-Vorgängen in Sekunden

qtime2 maximale Wartezeit in Asynchron-Vorgängen in Sekunden

Standard: 32767 (Sekunden)
Maximalwert: 32767 (Sekunden)
Minimalwert: 0 (Sekunden)

RECBUF=(number,length)

Größe des transaktionsgesicherten Wiederanlaufbereichs definieren. In den Wiederanlaufbereich werden Daten geschrieben, die für den Wiederanlauf nach Transaktions- oder Systemfehler benötigt werden. Zum Wiederanlaufbereich siehe Abschnitt „Wiederanlaufbereich“ auf Seite 51.

number Anzahl UTM-Seiten pro Prozess, die in der KDCFILE zur Aufnahme von Daten für den Wiederanlauf nach Systemfehler verwendet werden sollen. Die Größe einer UTM-Seite wird mit dem Operanden BLKSIZE= festgelegt. Wählt man diesen Bereich groß, wird die laufende Anwendung weniger belastet. Der Wiederanlauf nach Systemfehler dauert jedoch länger. Ist der Bereich klein, wird die laufende Anwendung stärker belastet, der Wiederanlauf nach Systemfehler ist jedoch schneller.

Standard: 100 (pro Prozess)
Minimalwert: 5 (pro Prozess)
Maximalwert: 32767 (pro Prozess)

length Größe des Puffers in Byte, der pro Prozess der Anwendung zur Zwischenspeicherung von Wiederanlaufdaten zur Verfügung steht. Die Daten werden für den Wiederanlauf nach Transaktions- oder Systemfehler benötigt.

Standard: 8192
Minimalwert: 1024
Maximalwert: 16777212 (16 MB)

RECBUFFS=number

Anzahl der Dateien, auf die der Wiederanlaufbereich aufgeteilt werden soll. Bei RECBUFFS=0 liegt der Wiederanlaufbereich in der Hauptdatei der KDCFILE. Zweite Exemplare bei doppelter Dateiführung (MAX ...,KDCFILE=(...,DOUBLE)) werden bei *number* nicht berücksichtigt. Die Dateinamen vergibt KDCDEF.

number darf nicht größer sein, als die in TASKS= angegebene maximale Prozessanzahl. Geben Sie für *number* einen größeren Wert an, so wird der Standardwert angenommen.

Standard: 0

B Maximalwert (BS2000/OSD): 99 bzw. Angabe bei MAX TASKS =
X/W Maximalwert (Unix-, Windows-Systeme):
X/W 10 bzw. Angabe bei MAX TASKS=

REDELIVERY=(number1, number2)

Maximale Anzahl wiederholter Zustellungen einer Asynchron-Nachricht, nachdem der Vorgang bzw. die Transaktion zurückgesetzt wurde. *number1* und *number2* gelten für unterschiedliche Nachrichtenziele.

number1 Maximale Anzahl wiederholter Zustellungen für Nachrichten an einen Asynchron-TAC. Die Zustellung wird immer dann wiederholt, nachdem ein Asynchron-Vorgang mit PEND ER/FR oder System PEND ER abnormal beendet wurde, ohne dass zuvor mindestens eine Transaktion erfolgreich abgeschlossen worden war. Ein erneuter Start eines Asynchron-Vorgangs nach PEND RS innerhalb der ersten Transaktion zählt nicht als Neuzustellung!

Bei einer erneuten Zustellung wird das diesem TAC zugeordnete Teilprogramm erneut gestartet.

Die Anzahl der erneuten Zustellungen wird beim FGET-Aufruf im KB-Rückgabebereich ausgegeben.

number2 Maximale Anzahl wiederholter Zustellungen für Nachrichten an eine Service-gesteuerte Queue. Die Zustellung wird immer dann wiederholt, nachdem die Nachricht verarbeitet und die Transaktion zurückgesetzt wurde.

Die Anzahl der erneuten Zustellungen wird beim DGET-Aufruf im KB-Rückgabebereich ausgegeben.

Standard: (0, 255)
Minimalwert für *number1* und *number2*: 0
Maximalwert für *number1* und *number2*: 255 (d.h. Anzahl nicht begrenzt)

Ein Wert 0 bedeutet, dass die Nachricht nach dem Rücksetzen, abhängig vom Wert in TAC ...,DEAD-LETTER-Q, gelöscht oder in der Dead Letter Queue gesichert wird.

Wird ein Wert auf 255 gesetzt, dann wird eine Nachricht ggf. beliebig oft zugestellt. Beachten Sie daher, dass ein solcher Wert zu einer Endlosschleife führen kann, z.B. wenn ein Teilprogramm wegen eines Programmierfehlers zurückgesetzt wird. Außerdem kann die Nachricht bei einer Endlosschleife nicht in der Dead Letter Queue gesichert werden.

B REQNR=number

B Maximale Anzahl der zu einer Zeit für eine Datei in einem UTM-Prozess parallel abgesetzten PAM-Schreib-/Leseaufträge. Mit diesem Wert kann die Parallelität von Ein- und Ausgaben in gewissen Grenzen beeinflusst werden.

B Standard: 2
B Minimalwert: 1
B Maximalwert: 100
B Ein Wert < 1 wird von KDCDEF ohne Meldung durch 1 ersetzt.

RESWAIT=(time1,time2)

(**res**ource **wait**)
Die für *time1* und *time2* angegebenen Zeiten können während des Betriebs der Anwendung mit dem Administrationskommando KDCAPPL geändert werden.

time1 Zeit in Sekunden, die ein Teilprogramm maximal auf ein von einer anderen Transaktion gesperrtes Betriebsmittel warten soll: GSSBs, TLSs, ULSs und in BS2000/OSD evtl. LTERM-Partner bei ANNOAMSG=N.
Ist nach dieser Zeit das Betriebsmittel nicht verfügbar, erhält das Teilprogramm einen entsprechenden Returncode.
Wartet die Transaktion, die das Betriebsmittel sperrt, auf eine Eingabe-Nachricht nach einem Programmaufruf PEND KP oder PGWT KP, so erhält das Teilprogramm den Returncode sofort, ohne Wartezeit *time1*. Beim PEND KP oder PGWT KP in einer Transaktion, die Betriebsmittel sperrt, werden alle wartenden Teilprogramme mit einem Returncode informiert.

RESWAIT=0 bedeutet, dass das Anwendungsprogramm nicht wartet. Ist ein benötigtes Betriebsmittel durch eine andere Transaktion gesperrt, erhält das anfordernde Teilprogramm sofort einen entsprechenden Returncode.

Standard: 120
Minimalwert: 0
Maximalwert: 32767

BB In BS2000/OSD ist die reale Wartezeit abhängig von der Genauigkeit, mit der die Börsenwartezeit beim Betriebssystem eingestellt wurde.

time2 Zeit in Sekunden, die maximal auf ein von einem anderen Prozess gesperrtes Betriebsmittel gewartet werden soll. Wird die maximale Wartezeit *time2* überschritten, dann beendet sich die Anwendung abnormal.

Sie sollten die Angabe für *time2* nicht zu klein wählen, denn bestimmte Aktivitäten in der UTM-Anwendung müssen von einem Prozess durchgeführt und abgeschlossen werden, bevor in einem anderen Prozess die gleichen Aktivitäten angestoßen werden können.

Beispiel

Ein Prozess sperrt beim Senden einer Nachricht das Terminal, für das die Nachricht bestimmt ist. Ein anderer Prozess, der eine Eingabe-Nachricht desselben Terminals bearbeiten will, muss warten.

Insbesondere muss die für *time2* angegebene Zeit mindestens so groß sein wie die längste Bearbeitungszeit (Realzeit) der im Folgenden beschriebenen Fälle:

– Bei einem Kommunikationspartner, der mit PTERM ...,PTYPE=APPLI generiert wurde, sind die Betriebsmittel für die Dauer eines Verarbeitungsschrittes gesperrt. Diese Zeit umfasst auch den Event-Exit VORGANG bei Vorgangsbeginn und/oder Vorgangsende.
– Bei Vorgangsende sind Betriebsmittel gesperrt, solange der Event-Exit VORGANG läuft.

Standard: 300
Minimalwert: 300
Maximalwert: 32767

Wird für *time2* der Wert 0 angegeben, nimmt KDCDEF ohne Meldung den Standardwert 300 an. Wird ein Wert zwischen 0 und 300 angegeben, gibt KDCDEF eine entsprechende Meldung aus.

B SAT= (**s**ecurity **a**udit **t**rail)
B Mindestprotokollierung von Ereignissen mit SAT
B Zu „SAT-Protokollierung“ siehe auch das openUTM-Handbuch „Einsatz von
B openUTM-Anwendungen unter BS2000/OSD“.

B ON Die SAT-Protokollierung wird eingeschaltet.

B Es wird eine Mindestprotokollierung mit SAT eingeschaltet für folgende Er-
B eignisse:
B – An- und Abmelden eines Prozesses bei der UTM-Anwendung
B – Umschalten des Speicherschutzschlüssels
B – Austausch von Programmen
B – Ausführung eines UTM-SAT-Administrationskommandos

B Die Mindestprotokollierung kann durch Preselection erweitert und gesteu-
B ert werden. Die Preselection wird generiert durch die Anweisung SATSEL
B und den Operanden SATSEL= in den Anweisungen USER und TAC. Durch
B das Administrationskommando KDCMSAT können die bei der Generierung
B eingestellten Preselection-Werte geändert werden.

B OFF Die SAT-Protokollierung wird ausgeschaltet.
B Es wird nur jeder Zugriff auf den SAT-Administrations-TAC KDCMSAT pro-
B tokolliert (außer KDCMSAT HELP). Alle anderen Ereignisse werden nicht
B protokolliert.

B Die SAT-Protokollierung kann mit dem SAT-Administrations-TAC KDCMSAT
B ein- und ausgeschaltet werden (siehe openUTM-Handbuch „Einsatz von
B openUTM-Anwendungen unter BS2000/OSD“).

B Standard: OFF

X/W SEMARRAY=(number,number1)

(**sem**aphore **array**)

Bereich von Semaphore Schlüssel für die anwendungsglobalen Semaphore (Prozesssynchronisation). Die Semaphore Schlüssel sind globale Parameter in Unix- bzw. Windows-Systemen. Mit SEMARRAY geben Sie einen Startwert *number* und eine Anzahl *number1* an. openUTM belegt diese Schlüssel, indem vom Startwert ausgehend jeweils 1 addiert wird. Näheres erfahren Sie bei Ihrem System-Administrator.

X/W

Die Parameter SEMARRAY= und SEMKEY= schließen sich gegenseitig aus. Der Vorteil von SEMARRAY= gegenüber SEMKEY= ist, dass mehr als zehn Semaphorschlüssel belegt werden können. Zur Berechnung der für eine UTM-Anwendung benötigten Semaphorschlüssel siehe folgende Beschreibung zu SEMKEY und Beschreibung der Systemglobalen Betriebsmittel im openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter Unix- und Windows-Systemen“.

X/W Pflichtoperand, wenn SEMKEY= nicht angegeben wurde.

X/W number Numerische Angabe des Startwertes

X/W number1 gibt die Anzahl von Schlüsseln an, die belegt werden sollen.

X/W Minimalwert: 1

X/W Maximalwert: 1000

X/W SEMKEY=(number,...)

(**sem**aphore **key**)

Semaphore Schlüssel für die anwendungsglobalen Semaphore (Prozess-Synchronisation). Die Semaphore Schlüssel sind globale Parameter in Unix- bzw. Windows-Systemen Es dürfen max. 10 Semaphore Schlüssel in einer Liste definiert werden. Alle Semaphore Schlüssel (*number,...*) werden als Dezimalzahlen angegeben. Näheres erfahren Sie bei Ihrem System-Administrator.

X/W Die Parameter SEMARRAY= und SEMKEY= schließen sich gegenseitig aus.

X/W Pflichtoperand, wenn SEMARRAY nicht angegeben wurde.

X/W

Die nachstehende Formel dient dazu, die Anzahl der für eine UTM-Anwendung benötigten Semaphor Schlüssel zu ermitteln. Der sich bei der Berechnung ergebende Wert ist auf eine ganze Zahl aufzurunden.

X/W Anzahl Semaphor Schlüssel = (9 + n + m) / 10

X/W Es bedeuten hierbei:

X/W – 9 = fester Wert, den openUTM für interne Synchronisationen benötigt.
X/W – n = maximale Anzahl während eines Anwendungslaufes gleichzeitig
X/W arbeitender Workprozesse.
X/W – m = maximale Anzahl während eines Anwendungslauf gleichzeitig
X/W angeschlossener externer Prozesse.

X/W Siehe auch: openUTM-Handbuch „Einsatz von openUTM-Anwendungen
X/W unter Unix- und Windows-Systemen“, Systemglobale Betriebsmittel.

SM2= legt fest, ob die UTM-Anwendung zur Überwachung der Performance Daten an SM2/openSM2 liefern soll.

NO Die Performanceüberwachung mit SM2/openSM2 wird für die UTM-Anwendung generell ausgeschlossen. D.h. die UTM-Anwendung kann keine Daten an SM2/openSM2 liefern, und der UTM-Administrator kann die Datenlieferung an SM2/openSM2 auch nicht einschalten.

OFF Die UTM-Anwendung kann Daten an SM2/openSM2 liefern. Der Administrator muss jedoch die Lieferung der Daten an SM2/openSM2 mit dem Administrationskommando KDCAPPL SM2=ON einschalten. Er kann die Lieferung der Daten auch jederzeit mit KDCAPPL=OFF wieder ausschalten.

Standardwert: OFF

ON Die UTM-Anwendung kann Daten an SM2/openSM2 liefern. Die Lieferung von Daten wird automatisch beim Start der UTM-Anwendung eingeschaltet. Der Administrator der UTM-Anwendung kann sie jederzeit mit dem Administrationskommando KDCAPPL SM2=OFF wieder ausschalten.

SPAB=length Maximale Länge des SPAB (Standard Primärer Arbeitsbereich) in Byte

Standard: 512
Minimalwert: 0
Maximalwert: 32767

STATISTICS-MSG=
legt fest, ob openUTM die Statistikmeldung K081 stündlich erzeugen soll oder nicht.

FULL-HOUR
Die Statistikmeldung K081 wird jede Stunde erzeugt und in die SYSLOG geschrieben. Gleichzeitig setzt openUTM folgende Anwendungs-spezifische Statistikwerte auf Null zurück:

– Anzahl der empfangenenen Nachrichten (*term_input_msgs*)
– Anzahl der gesendeten/ausgegebenen Nachrichten (*term_output_msgs*)
– Anzahl der Anforderungen, Sätze in die Benutzer-Protokolldatei USLOG zu schreiben (*logfile_writes*)

– Prozentualer Anteil der Anforderungen von Puffern im Cache, die zu Wartezeiten geführt haben (*cache_wait_buffer*)

NONE Die Statistikmeldung K081 wird nicht erzeugt und die oben angegebenen Statistikwerte werden nicht automatisch auf 0 zurückgesetzt.
NONE sollten Sie wählen, wenn Sie die oben aufgelisteten Statistikwerte bei Bedarf durch die Administration zurücksetzen wollen (siehe openUTM-Handbuch „Anwendungen administrieren“, KC_MODIFY_OBJECT).

Standard: FULL-HOUR

SYSLOG-SIZE=size

Automatische Größenüberwachung für die System-Protokolldatei SYSLOG durch openUTM generieren.

– *size*≠0
darf nur angegeben werden, wenn die System-Protokolldatei SYSLOG als Dateigenerationsgruppe (FGG) angelegt wird. Wird die SYSLOG als normale einfache Datei angelegt und für *size* ein Wert ≠ 0 angegeben, dann bricht openUTM den Start der Anwendung mit Startfehlercode 58 ab. Wird die SYSLOG als FGG angelegt, dann können Sie mit SYSLOG-SIZE die automatische Größenüberwachung der SYSLOG durch openUTM einschalten und mit *size* den Schwellwert für die Größe der Dateigenerationen festlegen, bei dem openUTM auf die nächste Dateigeneration umschaltet.

– *size*=0
Wird für *size* der Wert 0 eingegeben (Standard), dann überwacht openUTM die Größe der SYSLOG-Datei nicht, d.h. openUTM schreibt alle Meldungen mit dem Meldungsziel SYSLOG in dieselbe Dateigeneration, bis per Administration (Kommando KDCSLOG) auf eine andere Dateigeneration umgeschaltet oder die Größenüberwachung eingeschaltet wird.

– *size*≥100
Einen Wert ≥ 100 interpretiert openUTM wie folgt: Die Größe jeder einzelnen SYSLOG-Dateigeneration darf den Wert (*size* ∗ Größe einer UTM-Seite) nicht überschreiten. Die Größe einer UTM-Seite wird mit BLKSIZE definiert. openUTM schaltet automatisch auf die nächste SYSLOG-Dateigeneration um, wenn durch eine Meldungsausgabe in die SYSLOG die Größe der SYSLOG-Datei diesen Schwellwert überschreitet.

– *size*<100
Werte zwischen 1 und 99 werden von openUTM automatisch durch 100 ersetzt. In diesem Fall wird zur Information eine Meldung ausgegeben.

– *size*<0
Werte < 0 werden von KDCDEF zurückgewiesen.

Der UTM-Administrator kann den generierten Schwellwert verändern und die Größenüberwachung im Betrieb bei Bedarf ein- oder ausschalten (z.B. mit dem Administrationskommando KDCSLOG).

Standard: 0 (keine Größenüberwachung)
Minimalwert: 100
Maximalwert: (2^{31} - 1)

TASKS=number

Maximale Anzahl der Prozesse , die gleichzeitig für die Anwendung eingesetzt werden können.

Pflichtoperand.

Minimalwert: 2
Maximalwert: 250

Ein Wert < 2 wird von KDCDEF ohne Meldung durch 2 ersetzt.

Die aktuelle Anzahl der Prozesse wird beim Start der Anwendung festgelegt. Beim Start können Sie auch TASKS=1 angeben. Der Administrator kann dann die Anzahl der Prozesse im laufenden Betrieb dynamisch anpassen (z.B.mit dem Administrationskommando KDCAPPL).
Weder die beim Start angegebene Anzahl der Prozesse noch die vom Administrator eingestellte Anzahl darf den hier generierten Wert überschreiten.

TASKS-IN-PGWT=number

legt die maximale Anzahl der Prozesse der UTM-Anwendung fest, in denen gleichzeitig Teilprogramme mit blockierenden Aufrufen wie z.B. der KDCS-Aufruf PGWT ablaufen dürfen. Der Wert von TASKS-IN-PGWT muss kleiner sein als der Wert des Operanden TASKS.

Wird TASKS-IN-PGWT=0 angegeben, so kann keine TAC-Klasse und kein Transaktionscode (TAC) generiert werden, für die/den blockierende Aufrufe erlaubt sind (siehe TAC/TACCLASS ...,PGWT=). In diesem Fall muss in allen TACCLASS- und TAC-Anweisungen PGWT=NO angegeben werden, siehe hierzu auch die Anweisungen TAC auf Seite 488 und TACCLASS auf Seite 506.

Standardwert: 0
Minimalwert: 0
Maximalwert: *number* in TASKS -1

TERMWAIT=time

(**term**inal **wait**)

time Zeit in Sekunden, die in einer Mehrschritt-Transaktion (d.h. nach PEND KP) maximal zwischen einer Dialog-Ausgabe an den Partner und der anschließenden Dialog-Antwort vom Partner verstreichen darf. Dieser Wert gilt für alle Dialoge, in denen der Partner die Client-Rolle spielt (Terminals, UPIC-Clients, OSI TP-, LU6.1- und LU6.2-Auftraggeber). Bei Terminal-Clients z.B. entspricht *time* der Denkzeit des Benutzers nach PEND KP.
Bei Zeitüberschreitung wird die Transaktion abnormal beendet und die von ihr reservierten Betriebsmittel freigegeben. Die Verbindung zum Partner wird abgebaut.

Standard: 600
Maximalwert: 32767
Minimalwert: 60

TRACEREC=number

Maximale Anzahl der Einträge in den von openUTM geführten Prozess-spezifischen Trace-Bereichen. Dieser Wert gilt für die Trace-Bereiche

- der Main-Routine KDCROOT (UTM-DIAGAREA)
- des UTM-Systemcodes (KTA-Trace)
- des XAPTP-Bausteins (XAP-Trace) bei OSI TP-Anwendungen

openUTM schreibt in diese Bereiche Trace-Informationen für Diagnosezwecke.

Länge der Einträge:

- Eintrag in der UTM-DIAGAREA: 138 Byte (auf 32 Bit Systemen) bzw. 256 Byte (auf 64 Bit Systemen)
- KTA- bzw. XAP-Trace-Eintrag: 64 Byte (auf 32 Bit Systemen) bzw. 112 Byte (auf 64 Bit Systemen)

Standard: 32500
Minimalwert: 1
Maximalwert: max. 32500, in Abhängigkeit vom verfügbaren Speicher

Ein Wert < 1 wird von KDCDEF ohne Meldung durch den Standardwert ersetzt, ein Wert > 32500 durch den Maximalwert.

TRMSGLTH=length

Legt den Maximalwert fest für:

- Die Länge der physikalischen Ausgabe-Nachricht, die an ein Terminal, einen Drucker oder eine Transportsystem-Anwendung (PTYPE=APPLI) gesendet wird. Bei der Längenberechnung müssen alle zu übertragenden Zeichen mitgerechnet werden, auch Steuerzeichen etc.
- Die Länge der asynchronen Ausgabe-Nachrichten an Transportsystem-Anwendungen vom Typ SOCKET.

Standard: 4096 Byte
Minimalwert: 4096 Byte
Maximalwert: 32700 Byte

X/W X/W X/W In Unix- und Windows-Systemen muss der in *length* angegebene Wert mindestens 24 Byte größer sein als der beim Operanden NB angegebene Wert.

B B B B Wenn RSO-Drucker verwendet werden, muss die Größe des RSO-Puffers (REMOTE-BUFFER-SIZE in der SPOOL-Parameterdatei) größer gleich TRMSGLTH=*length* sein. Siehe hierzu auch Abschnitt „RSO-Puffergröße festlegen“ auf Seite 173.

USLOG= legt die einfache oder doppelte Dateiführung für die Benutzer-Protokolldatei USLOG fest.

SINGLE Die Benutzer-Protokolldatei soll einfach geführt werden.

Standard: SINGLE

DOUBLE Die Benutzer-Protokolldatei soll aus Sicherheitsgründen doppelt geführt werden. Näheres zur Benutzer-Protokolldatei finden Sie im openUTM-Handbuch „Einsatz von openUTM-Anwendungen“.

B VGMSIZE=number
B B B B Für das Vorgangs-Memory eines SQL-Datenbanksystems können Sie mit diesem Parameter einen Pufferbereich in der angegeben Größe generieren. Damit wird auch der Anteil eines Benutzers am Pagepool begrenzt. VGMSIZE= wird in KB angegeben.

B B Ist der zum PEND-Zeitpunkt aktuell zu sichernde VGM-Bereich größer als *number*, wird der Vorgang mit PEND ER abgebrochen.

B B B Standardwert: 32 KB
Minimalwert: 32 KB
Maximalwert: 256 KB

X/W XAPTPSHMKEY=number
X/W Schlüssel für das XAPTP Shared Memory Segment.

X/W Die Schlüssel sind globale Parameter in Unix- bzw. Windows-Systemen.

X/W X/W XAPTPSHMKEY ist Pflichtoperand, falls die Anwendung über OSI TP-Protokoll kommuniziert.

Die folgende Tabelle zeigt den Verwendungszweck der einzelnen Operanden der MAX-Anweisung sowie deren Standardwerte im Überblick:

	Operand	Verwendungszweck	Pflicht	Standardwert
	betriebssystemunabhängige Operanden			
	APPLIMODE=	Wahl einer UTM-Variante: UTM-S oder UTM-F		SECURE
	APPLINAME=	Name der UTM-Anwendung	X	-
	ASYNTASKS=	Asynchron-Verarbeitung (Anzahl Prozesse für Asynchron-Verarbeitung und gleichzeitig offener Asynchron-Vorgänge)		1, 1
	BLKSIZE=	Größe einer UTM-Seite festlegen		2K
B B B X/W X/W	CACHESIZE=	Tuningmaßnahme (Größe und Eigenschaften des Cache-Speichers)		Systemabhängig: BS2000/OSD: (1024,70%, NORES) Unix-/Windows-Systeme: (1024,70%)
	CLRCH=	Zeichen zum Überschreiben von KB und SPAB		keines
B B B X/W X/W X/W	CONN-USERS=	Anzahl der gleichzeitig aktiven Benutzer bzw. Clients beschränken		Systemabhängig: BS2000/OSD: keine Beschränkung Unix- und Windows-Systeme: Pflichtoperand
	CONRTIME=	automatischer Verbindungsaufbau (Wartezeit für Wiederaufbau einer Verbindung)		10 Min
	DEAD-LETTER-Q-ALARM=	Überwachung des Nachrichten-Aufkommens in der Dead Letter Queue		0, d.h. keine Überwachung
	DESTADM=	asynchrone Administration		keine
	DPUTLIMIT1=	zeitgesteuerte Aufträge (obere Schranke)		360 Tage
	DPUTLIMIT2=	zeitgesteuerte Aufträge (untere Schranke)		1 Tag
	GSSBS=	GSSB-Speicherbereiche (Maximal-Anzahl)		32
	HOSTNAME=	virtueller Hostname für die UTM-Anwendung		8 Leerzeichen
	KB=	maximale Länge des Kommunikationsbereich		512
	KDCFILE=	KDCFILE zuordnen	X	-
	KEYVALUE=	Zugriffsschutz nach dem Lock-Keycode-Konzept (Nummer des größten Keycodes)		32

Operand	Verwendungszweck	Pflicht	Standardwert
LEADING-SPACES=	Führende Leerzeichen in Nachrichten von Terminals oder von TS-Anwendungen (PTYPE=APPLI/SOCKET) an das Teilprogramm weiterreichen		NO
LPUTBUF=	Protokollierung von Benutzerdaten mit LPUT (Anzahl PAM-Seiten im Pagepool)		1
LPUTLTH=	Protokollierung von Benutzerdaten mit LPUT (Maximale LPUT-Nachrichtenlänge)		1948 Byte
LSSBS=	LSSB-Speicherbereiche (maximale Anzahl)		8
NB=	maximale Länge des Nachrichtenbereichs		2048
NRCONV=	Maximalzahl gekellerter Vorgänge		0
OSI-SCRATCH-AREA=	Größe für UTM-internen Arbeitsbereich in KB		256
PGPOOL=	Pagepool-Größe und Warnstufen		100 UTM-Seiten, 80%, 95%
PGPOOLFS=	Tuningmaßnahme: Aufteilung Pagepool		Pagepool in KDCFILE
PGWTTIME=	maximale Zeit für den KDCS-Aufruf PGWT		TERMWAIT=*time*
QTIME	Maximale Zeit für das Warten auf Nachrichten von Service-gesteuerten Queues festlegen		32767 Sekunden
RECBUF=	Tuningmaßnahme: Größe Wiederanlaufbereich in KDCFILE bzw. Prozess-spezifischem Systemspeicher		100 UTM-Seiten pro Prozess, 8192 Byte
RECBUFFS=	Tuningmaßnahme: Aufteilung Wiederanlaufbereich		Wiederanlaufbereich in KDCFILE
REDELIVERY=	Maximale Anzahl wiederholter Zustellungen einer Asynchron-Nachricht festlegen		0 für UTM-gesteuerte Queues, 255 für Service-gesteuerte Queues
RESWAIT=	Wartezeit auf ein durch eine andere Transaktion (*time1*) bzw. durch einen anderen Prozess (*time2*) gesperrtes Betriebsmittel (z.B. GSSB, TLS)		120 sek 300 sek
SPAB=	maximale SPAB-Länge		512
STATISTICS-MSG=	Statistikmeldung K081 erzeugen und Zählerstände automatisch zurücksetzen lassen		FULL-HOUR
SYSLOG-SIZE=	automatische Größenüberwachung der SYSLOG-Datei durch openUTM festlegen		0

	Operand	Verwendungszweck	Pflicht	Standardwert
	TASKS=	Anzahl UTM-Prozesse	X	-
	TASKS-IN-PGWT=	Anzahl der Prozesse für blockierende Aufrufe		0
	TERMWAIT=	maximale Wartezeit auf eine Dialog-Eingabe innerhalb einer Transaktion		600 sek
	TRACEREC=	Platz reservieren für Diagnoseinformation (Anzahl der Einträge)		32500
	TRMSGLTH=	maximale Nachrichtenlänge		4096 Byte
	USLOG=	Benutzer-Protokolldatei einfach oder doppelt führen		einfach
	VGMSIZE=	Pufferbereich in der angegeben Größe generieren		32 KB
B	***BS2000/OSD-spezifische Operanden***			
B B	BRETRYNR=	Kommunikation mit BCAM (Anzahl Versuche der Nachrichtenübergabe)		10,0
B	CARDLTH=	Ausweisleser für KDCSIGN-Prüfung		0
B	CATID=	Katalogkennungen für die KDCFILE festlegen		Standard-CATID
B	LOCALE=	Standard-Sprachumgebung festlegen		Leerzeichen
B B	LOGACKWAIT=	Unterstützung von Ausgabegeräten (Wartezeit auf Quittung)		600 sek
B B	MP-WAIT=	maximale Wartezeit pro Prozess für Anschluss an Common Memory Pool		180 sek
B B	PRINCIPAL-LTH=	maximale Länge eines Kerberos-Principals in Bytes		32
B B	REQNR=	Tuningmaßnahme: PAM-I/O-Aufträge (maximale Anzahl paralleler Aufträge)		2
B B	SAT=	Mindestprotokollierung von Ereignissen mit SAT		OFF
B B	SM2=	Lieferung der UTM-Daten an SM2 zulassen und ein-/ausschalten		OFF
B B	VGMSIZE=	Größe des Pufferbereichs für das Vorgangsgedächtnis eines SQL-Datenbanksystems		32KB

	Operand	Verwendungszweck	Pflicht	Standardwert
X/W	*Unix- und Windows-System-spezifische Operanden*			
X/W X/W	CACHESHMKEY=	Schlüssel für Shared Memory Segment (anwendungsglobale Puffer für Dateizugriffe)	X	-
X/W X/W	IPCSHMKEY=	Schlüssel für Shared Memory Segment (Kommunikation zwischen UTM-Prozessen)	X	-
X/W	IPCTRACE=	Anzahl UTM-Einträge in IPC-Trace festlegen		1060
X/W X/W	KAASHMKEY=	Schlüssel für Shared Memory Segment (anwendungsglobale Daten)	X	-
X/W X/W	NET-ACCESS=	Abbildung Netzverbindungen auf Netzprozesse		MULTI-THREADED
X/W X/W	OSISHMKEY=	Schlüssel für Shared Memory Segment von OSS	bei OSI TP	-
X/W X/W X/W	SEMARRAY=	Bereich von Semaphore Schlüssel für anwendungsglobale Semaphore (alternativ zu SEMKEY)	X	-
X/W X/W	SEMKEY=	Semaphore Schlüssel für anwendungsglobale Semaphore (alternativ zu SEMARRAY)	X	-
X/W X/W	XAPTPSHMKEY=	Schlüssel für das XAPTP Shared Memory Segment	bei OSI TP	-

MESSAGE - Meldungsmodul beschreiben

Mit der MESSAGE-Anweisung werden Benutzermeldungsmodule in die Konfiguration aufgenommen. Mit einem eigenen Benutzermeldungsmodul können Sie die Meldungstexte und/oder die Meldungsziele einzelner Meldungen an Ihre Bedürfnisse anpassen.

Zum Thema Meldungsmodule siehe auch den Abschnitt „UTM-Meldungen“ auf Seite 183 in diesem Handbuch und das openUTM-Handbuch „Meldungen, Test und Diagnose“.

B **Meldungsmodule in BS2000/OSD generieren**

B Zur Internationalisierung der Anwendung können mehrere Benutzermeldungsmodule erB zeugt werden, die die UTM-Meldungen einer Anwendung dann in der jeweiligen LandesB sprache ausgeben.

B Für Benutzermeldungsmodule kann die jeweilige Sprachumgebung über ein Locale mit B einem eindeutigen Paar „Sprach- und Territorialkennzeichen“ spezifiziert werden. Die B Zuordnung der landessprachlichen Meldungsmodule für die Meldungausgabe erfolgt entB sprechend dem Locale (passende Kombination von Sprach- und Territorialkennzeichen) B der Benutzer und LTERM-Partner.

B Mit openUTM wird standardmäßig das deutsche Meldungsmodul KCSMSGS und das engB lische Standardmeldungsmodul KCSMSGSE ausgeliefert.

```
B  MESSAGE␣      MODULE=name
B                [ ,LIB=onlname ]
B                [ ,LOCALE = (lang-id [,terr-id]) ]
```

B MODULE=name

Name des Benutzermeldungsmoduls. *name* kann max. 8 Zeichen lang sein.
B Das Modul wird mit dem Tool KDCMMOD erzeugt (siehe openUTM-HandB buch „Meldungen, Test und Diagnose“).

B Pflichtoperand.

B Der Name des Meldungsmoduls muss in der Anwendung eindeutig sein.

B B LIB=omlname bezeichnet die Objektmodulbibliothek, aus der das Benutzermeldungsmodul nachgeladen werden soll. *omlname* kann max. 54 Zeichen lang sein.

B B Falls das Benutzermeldungsmodul nachgeladen werden soll, darf es nicht zur Anwendung dazugebunden werden.

B B B B B Geben Sie LIB= nicht an, dann wird LIB= mit TASKLIB vorbelegt. Dies entspricht **nicht** dem Kommando SET-TASKLIB, sondern es muss dann eine Bibliothek mit dem Namen TASKLIB existieren. Ein Nachladen des Benutzermeldungsmoduls aus der mit SYSFILE-TASKLIB zugewiesenen Bibliothek wird nicht unterstützt.

B B B B B B **i** Beim Nachladen sucht der DBL das Benutzermeldungsmodul zuerst in der Bibliothek, die Sie in LIB= zugewiesen haben. Existiert die Bibliothek nicht, dann bricht der DBL die Suche ab. Ist die Bibliothek zwar vorhanden, das Benutzermeldungsmodul wird dort aber nicht gefunden, dann durchsucht der DBL die alternativen Bibliotheken. Das sind die Bibliotheken, denen ein Dateikettungsname BLSLIB*nn* (0≤*nn*≤99) zugewiesen wurde.

B B B B B B B LOCALE=(lang_id, terr_id)
definiert über ein Sprachkennzeichen und ggf. ein Territorialkennzeichen die Sprachumgebung der Benutzer-spezifischen Meldungsmodule. Über die entsprechenden Angaben für den Parameter LOCALE= in der USER- bzw. LTERM-Anweisung erfolgt die Zuordnung des Meldungsmoduls und die Meldungen werden in der jeweiligen Landessprache des Benutzers ausgegeben.

B B B B Wird mehr als eine MESSAGE-Anweisung angegeben, müssen alle den Parameter LOCALE= enthalten. Dabei darf in der MESSAGE-Anweisung für ein Meldungsmodul nicht eine bereits verwendete Kombination aus *lang_id* und *terr_id* angegeben werden.

B B lang_id Max. zwei Zeichen langes Sprachkennzeichen für ein Meldungsmodul. Das Sprachkennzeichen ist frei wählbar.

B B Für *lang_id* existiert kein Standardwert, d.h. es muss immer ein Wert für *lang_id* angegeben werden.

terr_id

Max. zwei Zeichen langes Territorialkennzeichen für ein Meldungsmodul. Für *terr_id* können auch Leerzeichen angegeben werden.

Wenn Sie MESSAGE ...,LOCALE= angegeben, müssen Sie auch MAX ...,LOCALE= versorgen (siehe Seite 385). Anwendungs-Meldungsmodul der UTM-Anwendung wird automatisch das Meldungsmodul, dessen *lang_id* und *terr_id* in der MESSAGE-Anweisung mit dem Locale in der MAX-Anweisung übereinstimmen. Das Anwendungs-Meldungsmodul wird von openUTM für Meldungen an SYSLST, SYSOUT und CONSOLE verwendet. Die Angaben zu den Meldungszielen in den anderen Meldungsmodulen sind ohne Bedeutung.

Für Meldungen an STATION, SYSLINE und PARTNER wird das Meldungsmodul verwendet, dessen *lang_id* und *terr_id* in der MESSAGE-Anweisung mit den Angaben für LOCALE= in der USER- bzw. LTERM-Anweisung identisch sind.

Hierbei dominieren die Angaben zum Benutzer die zum LTERM-Partner, d.h. ist zum Zeitpunkt der Meldungsausgabe ein Benutzer angemeldet, so verwendet openUTM das zum Benutzer passende Meldungsmodul. Bei der Zuordnung der Meldungsmodule anhand der Sprach- und Territorialkennzeichen wird wie folgt verfahren:

- Existiert ein Meldungmodul mit der Kombination *lang_id* und *terr_id*, die mit den Angaben in der USER- bzw. LTERM-Anweisung identisch ist, so werden die Meldungen in dieser Sprachumgebung ausgegeben.
- Ist keine identische Kombination vorhanden, wird das Meldungsmodul verwendet, das in *lang_id* übereinstimmt und für das *terr_id* nicht generiert wurde.
- Wenn auch dies nicht möglich ist, wird das Anwendungs-Meldungsmodul verwendet.

X/W **Meldungsmodul in Unix- oder Windows-Systemen generieren**

X/W In Unix- und Windows-Systemen können Sie mit der MESSAGE-Anweisung genau ein benutzereigenes Meldungsmodul generieren, d.h. pro KDCDEF-Lauf dürfen Sie die MESSAGE-Anweisung nur einmal angeben.

X/W Wird keine MESSAGE-Anweisung angegeben, so heißt der Name der externen C/C++-Struktur KCSMSGS. Ein Objektmodul mit einer C/C++-Struktur dieses Namens wird als Datei mit openUTM ausgeliefert.

X In Unix-Systemen ist dies das Objektmodul `kcsmsgs.o` in der Bibliothek
X *utmpfad*`/sys/libwork.a` bzw. *utmpfad*`/sys/libwork.so`.

W In Windows-Systemen ist es das Modul `kcsmsgs.obj` im Verzeichnis
W *utmpfad*`\sys`.

X/W
```
MESSAGE␣      MODULE=name
```

X/W MODULE=name

X/W gibt den Namen der externen C/C++-Struktur an, über die die Meldungen adressiert werden. Wird ein Benutzermeldungsmodul verwendet (siehe Tool KDCMMOD im openUTM-Handbuch „Meldungen, Test und Diagnose“), dann muss *name* mit dem Namen dieses Moduls übereinstimmen. *name* kann max. 8 Zeichen lang sein.

X/W Pflichtoperand.

MPOOL - Common Memory Pool beschreiben (BS2000/OSD)

B Mit dieser Anweisung werden die Eigenschaften eines Common Memory Pools festgelegt.

B Die Anweisung darf mehrmals angegeben werden. Die Obergrenze hängt davon ab, wie viele dieser Pools von einem Prozess angelegt werden können. Es werden jeweils maximal 8 Common Memory Pools mit SCOPE=GROUP bzw. SCOPE=GLOBAL unter einer Benutzerkennung unterstützt.

B Die Common Memory Pools werden immer FIXED angelegt. Jedem Task, der sich an einem bereits existierenden Common Memory Pool anschließt, wird die gleiche Adresse zugewiesen wie dem Task, der den Common Memory Pool eingerichtet hat.

B Die Reihenfolge der MPOOL-Anweisungen innerhalb des Generierungslaufs beeinflusst die Reihenfolge, in der die Common Memory Pools eingerichtet werden. Zuerst werden alle Common Memory Pools eingerichtet, die mit SCOPE=GLOBAL generiert sind, in der Reihenfolge der MPOOL-Anweisungen innerhalb des Generierungslaufs. Danach werden alle Common Memory Pools eingerichtet, die mit SCOPE=GROUP generiert sind, in der Reihenfolge der MPOOL-Anweisungen innerhalb des Generierungslaufs.

```
MPOOL␣      poolname
            [ ,ACCESS={ READ | WRITE } ]
            [ ,PAGE=X'xxxxxxxx' ]
            [ ,SCOPE={ GROUP | GLOBAL } ]
              ,SIZE=poolsize
```

poolname — Name des Common Memory Pools. *poolname* muss innerhalb der UTM-Anwendung eindeutig sein und kann max. 50 Zeichen lang sein.

Es wird eine Ziffer an den Namen angefügt.

ACCESS= — definiert die Zugriffsberechtigung.

READ — Auf den Common Memory Pool kann nur lesend zugegriffen werden.

Standard: READ

WRITE — Auf den Common Memory Pool kann sowohl lesend als auch schreibend zugegriffen werden.

PAGE=X'xxxxxxxx'

Hexadezimale Adresse in der Form X'xxxxxxxx'.

– 24-Bit-Adressierungsmodus:
  Ist die Adresse nicht ein Vielfaches von 64 KB (die niederwertigen vier Halb-Bytes sind 0), dann wird die angegebene Adresse auf ein Vielfaches von 64 K aufgerundet.

– 31-Bit-Adressierungsmodus:
  Die Adresse ist ein Vielfaches von 1 MB. Ist dies nicht der Fall, so wird die Adresse auf ein Vielfaches von 1 MB aufgerundet.

Standard:

– 24-Bit-Adressierungsmodus:
  Der Pool wird ab der niedrigst möglichen Adresse angelegt.

– 31-Bit-Adressierungsmodus:
  Der Pool wird ab der niedrigst möglichen Adresse über X'01000000' angelegt.

i Werden im BS2000 globale Common Memory Pools gleichen Inhalts/Namens in mehreren UTM-Anwendungen verwendet, muss der Parameter PAGE=X'xxxxxxxx' mit gleicher Adresse in allen Anwendungen angegeben werden. Die mit PAGE= angegebene Adresse ist so zu wählen, dass der damit reservierte Adressbereich in all diesen Anwendungen verfügbar ist.

Die Common Memory Pools werden immer FIXED angelegt, d.h. alle Tasks der UTM-Anwendung finden den Pool auf der gleichen Adresse ihres virtuellen Adressraumes.

Eine Alternative für die Verwendung von PAGE= ist, die gemeinsamen Pools in allen Anwendungen in der gleichen Reihenfolge zu generieren. Die MPOOL-Anweisungen für die gemeinsamen Pools müssen dabei am Anfang der MPOOL-Anweisungen stehen.

SCOPE= definiert den Geltungsbereich des Common Memory Pools.

GLOBAL definiert einen Geltungsbereich für alle Prozesse im System.

GROUP definiert einen Geltungsbereich für alle Prozesse, die unter derselben Benutzerkennung laufen.

Standard: GROUP

B B SIZE=poolsize

Anzahl der 64 KB großen Speicherabschnitte des Pools (1 Einheit ≙ 64 KB)

B B B Im 31-Bit-Adressierungsmodus sind die Speicherabschnitte 1MB groß. Die Größe des Common Memory Pools wird dann auf die nächste 1MB-Grenze aufgerundet, die sich durch *poolsize* mit 64KB multipliziert ergibt.

B Pflichtoperand

MSG-DEST - Benutzer-spezifische Meldungsziele definieren

Mit dieser Anweisung können Sie bis zu vier zusätzliche Benutzer-spezifische Meldungsziele für die UTM-Meldungen definieren.

Dazu stellt openUTM die frei verfügbaren UTM-Meldungsziele USER-DEST-1, USER-DEST-2, USER-DEST-3 und USER-DEST-4 zur Verfügung. Mit MSG-DEST können Sie diesen UTM-Meldungszielen konkrete Ziele zuweisen. Diese Ziele können sein:

- eine USER-Queue, d.h. die Message Queue einer Benutzerkennung
- eine TAC-Queue
- ein Asynchron-TAC
- oder ein LTERM-Partner, der keinem UPIC-Client zugeordnet ist.

Sie können auch mehrere Meldungsziele vom gleichen Typ zuweisen, z.B. drei LTERM-Partner und eine USER-Queue. Durch die Definition einer USER- oder TAC-Queue als Benutzer-spezifisches Meldungsziel können Sie auch erreichen, dass UTM-Meldungen an den WinAdmin-Administrationsarbeitsplatz ausgegeben werden. Näheres hierzu finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose“.

```
MSG-DEST␣     msgdest
              ,NAME=name
              ,DEST-TYPE={ LTERM | USER-QUEUE | TAC }
              [ ,MSG-FORMAT={ FILE | PRINT } ]
```

msgdest

Name des UTM-Meldungsziels, dem Sie ein Benutzer-spezifisches Meldungsziel zuweisen möchten. Mögliche Werte sind:

USER-DEST-1, USER-DEST-2, USER-DEST-3 oder USER-DEST-4.

msgdest müssen Sie außerdem mit Hilfe von KDCMMOD denjenigen Meldungen zuordnen, die an dieses Benutzer-spezifische Meldungsziel ausgegeben werden sollen. Näheres siehe Abschnitt „Benutzer-spezifische Meldungsziele“ auf Seite 188 sowie die Beschreibung von KDCMMOD im openUTM-Handbuch „Meldungen, Test und Diagnose“.

NAME=name Name des Benutzer-spezifischen Meldungsziels, mögliche Angaben:

- Name einer UTM-Benutzerkennung. Diese muss in einer USER-Anweisung generiert sein.
- Name eines Asynchron-TACs. Dieser muss in einer TAC-Anweisung mit TYPE=A generiert sein.
- Name einer TAC-Queue. Diese muss in einer TAC-Anweisung mit TYPE=Q generiert sein.

B
B
B
B
X/W
X/W
X/W
X/W

- *BS2000/OSD*:
  Name eines LTERM-Partners. Dieser muss in einer LTERM-Anweisung generiert sein und darf keinem PTERM mit PTYPE=UPIC-R zugeordnet sein.
- *Unix- und Windows-Systeme*:
  Name eines LTERM-Partners. Dieser muss in einer LTERM-Anweisung generiert sein und darf keinem PTERM mit PTYPE=UPIC-R oder UPIC-L zugeordnet sein.

Alle Meldungen, die über KDCMMOD mit *msgdest* verknüpft wurden, werden dann auch an das in *name* angegebene Ziel ausgegeben.

Benutzer-spezifische Meldungsziele sollten weder gesperrt noch dynamisch gelöscht werden, da dann keine Meldungen mehr an dieses Ziel ausgegeben werden.

DEST-TYPE= gibt den Typ des in *name* angegebenen Meldungsziels an:

LTERM Das in *name* angegebene Meldungsziel ist ein LTERM-Partner.

TAC Das in *name* angegebene Meldungsziel ist ein Asynchron-TAC oder eine TAC-Queue.

USER-QUEUE
Das in *name* angegebene Meldungsziel ist eine USER-Queue.

MSG-FORMAT=
gibt das Format an, in dem die Meldung an das Meldungsziel übergeben wird.

FILE Das Format entspricht den Datenstrukturen für das MSGTAC-Programm. D.h. es werden nur die Meldungsinserts ohne den Meldungstext übergeben und die Meldungsinserts werden nicht in ein abdruckbares Format konvertiert.

PRINT Das Format entspricht dem Ausgabeformat des UTM-Tools KDCPSYSL. D.h. der Meldung werden Datum und Uhrzeit vorangestellt, danach folgen der Meldungstext mit den Text-Inserts und den zusätzlichen Inserts. Alle Inserts sind abdruckbar.

KDCPSYSL ist im openUTM-Handbuch „Einsatz von openUTM-Anwendungen“ beschrieben.

Standard: FILE

MUX - Multiplexanschluss definieren (BS2000/OSD)

B Mit dieser Anweisung werden Namen und Eigenschaften einer Multiplexverbindung zwischen der UTM-Anwendung und einem Session Manager (OMNIS) definiert. Über diesen Multiplexanschluss können sich mehrere Terminals gleichzeitig an die UTM-Anwendung anschließen.

B Die Initiative zum Aufbau der Transportverbindung zwischen openUTM und dem Session Manager kann sowohl vom Session Manager als auch von openUTM ausgehen, die Initiative zum Session-Aufbau aber nur vom Session Manager.

```
MUX␣          name
              [ ,BCAMAPPL=local_appliname ]
              [ ,CONNECT={ Y | N } ]
              [ ,MAXSES=number ]
              [ ,NETPRIO={ MEDIUM | LOW } ]
                ,PRONAM={ processorname | C'processorname' }
              [ ,STATUS={ ON | OFF} ]
```

name Name des Multiplexanschlusses

Der angegebene Name muss eindeutig sein und darf keinem weiteren Objekt der Namensklasse 3 zugeordnet sein. Siehe dazu auch Abschnitt „Eindeutigkeit der Namen und Adressen“ auf Seite 262.

BCAMAPPL=local_appliname
Lokaler Anwendungsname der UTM-Anwendung, wie er in der MAX-Anweisung (APPLINAME auf Seite 372) oder mit einer BCAMAPPL-Anweisung, siehe Seite 289, festgelegt wurde. Die Verbindung zum Session Manager wird über diesen Anwendungsnamen aufgebaut, d.h. der Session Manager muss beim Verbindungsaufbau zur UTM-Anwendung *local_appliname* als Partnernamen angeben. Werden für einen Session Manager mehrere MUX-Anweisungen mit verschiedenen BCAMAPPL-Namen angegeben, dann können zu diesem Session Manager parallele Verbindungen aufgebaut werden.

Der in der CLUSTER-Anweisung angegebene BCAMAPPL-Name ist hier nicht erlaubt.

Standard:
Anwendungsname der Anweisung MAX APPLINAME=*appliname*

CONNECT= Aufbau der logischen Transportverbindung bei Anwendungsstart

Y openUTM versucht beim Start der Anwendung eine logische Transportverbindung zum Session Manager aufzubauen.

Kommt keine Verbindung zustande, wiederholt openUTM den Versuch, die Verbindung aufzubauen und zwar in dem Zeitabstand, der mit MAX ...,CONRTIME=*time* definiert wurde.

Standard: Y

N openUTM versucht nicht, beim Start der Anwendung eine Verbindung zum Session Manager aufzubauen.

MAXSES=number

Maximale Anzahl gleichzeitig aktiver Sessions zwischen dem Session Manager und der UTM-Anwendung.

openUTM erzeugt für *number* Sessions intern auch *number* LTERM-Partner. Die Anzahl dieser LTERM-Partner muss bei den Maximalwerten für UTM-Namen berücksichtigt werden. Siehe Abschnitt „Maximalwerte für Namen“ auf Seite 260.

Standardwert: 10
Minimalwert: 1
Maximalwert: 65000 (theoretischer Wert)

NETPRIO= definiert die Transportpriorität, die auf der Transportverbindung zwischen Session Manager und UTM-Anwendung benutzt werden soll.

Standard: MEDIUM

PRONAM={ processorname | C’processorname’ }

Name des Rechners, auf dem sich der Session Manager befindet. Enthält *processorname* Sonderzeichen, muss er als Zeichenkette mit C’...’ angegeben werden.

STATUS= Gibt den Status des Multiplexanschlusses an.

ON Der Anschluss an den Session Manager ist nicht gesperrt.

Standard: ON

OFF Der Anschluss an den Session Manager ist gesperrt. Es kann keine Verbindung zwischen Session Manager und UTM-Anwendung aufgebaut werden.

Der Status kann vom Administrator geändert werden.

OPTION - KDCDEF-Lauf steuern

Mit der Steueranweisung OPTION kann der KDCDEF-Lauf gesteuert werden. Dabei werden aus einer Prozedurdatei/Shellscript ausschließlich die Optionen für den KDCDEF-Lauf eingelesen, während die eigentlichen Generierungsanweisungen aus weiteren Dateien eingelesen werden (unter BS2000/OSD aus SAM- oder ISAM-Dateien).

openUTM liest OPTION-Anweisungen nur von SYSDTA bzw. *stdin*.
OPTION-Anweisungen werden von KDCDEF ignoriert, wenn sie aus einer Datei gelesen werden, die mit OPTION DATA= zugewiesen wurde.

Wird die OPTION-Anweisung mehrfach angegeben, so sind jeweils die zuletzt gemachten Angaben gültig.

Wird keine OPTION-Anweisung angegeben, dann erzeugt KDCDEF nur die KDCFILE, d.h. es gilt die Voreinstellung GEN=KDCFILE.

```
OPTION␣       [ ,DATA=filename ]

              [ ,GEN= { KDCFILE | ROOTSRC | NO | ALL  |
                        CLUSTER                       |
                       (KDCFILE,ROOTSRC)              |
                       (CLUSTER,KDCFILE)              |
                       (CLUSTER,ROOTSRC)              |
                       (CLUSTER,KDCFILE,ROOTSRC) } ]

              [ ,GEN-RSA-KEYS={ YES | NO } ]
```

B ***BS2000/OSD-spezifischer Operand***

B `[ ,ROOTSRC=filename ]`

X/W ***Unix- und Windows-System-spezifische Operanden***

X/W `[ ,CHECK-RFC1006={ NO | YES } ]`

X/W CHECK-RFC1006=
X/W Erweiterte Überprüfung der UTM-Generierung für die Kommunikation über
X/W TCP/IP-Verbindungen mit RFC1006

X/W YES KDCDEF überprüft die Angaben zur Transportadresse für alle Kommunika-
X/W tionspartner und lokale Transportsystemendpunkte, die mit T-PROT=
X/W RFC1006 generiert sind, auf Vollständigkeit und Plausibilität. Bei OPTION
X/W CHECK-RFC1006=YES muss bei den Parametern LISTENER-PORT der
X/W Anweisungen ACCESS-POINT, BCAMAPPL, CON, OSI-CON und PTERM
X/W eine Portnummer angegeben werden.
X/W Standard: YES

X/W NO KDCDEF führt keine erweiterte Prüfung durch.

DATA=filename

Die KDCDEF-Steueranweisungen werden aus der hier spezifizierten Datei (unter BS2000/OSD aus einer SAM- oder ISAM-Datei) gelesen. Wird das Dateiende erreicht, dann werden die nächsten KDCDEF-Steueranweisungen wieder von SYSDTA bzw. *stdin* gelesen.

Mit DATA=*control_statements_file* können Sie KDCDEF Inputdateien zuweisen, die im inversen KDCDEF mit der Anweisung CREATE-CONTROL-STATEMENTS erzeugt wurden. Die Inputdateien *control_statements_file* enthalten aus der aktuellen KDCFILE ausgelesene Konfigurationsinformationen, die in KDCDEF-Steueranweisungen umgewandelt wurden.

Zur Funktion inverser KDCDEF siehe auch Abschnitt „Inverser KDCDEF“ auf Seite 265.

GEN= gibt an, welche Objekte erzeugt werden sollen.

KDCFILE Es wird die KDCFILE erzeugt.

Für eine UTM-Cluster-Anwendung wird außerdem die Cluster-User-Datei erzeugt bzw. eine bereits existierende Cluster-User-Datei ausgewertet und gegebenenfalls erweitert.

Standard: KDCFILE

ROOTSRC

Es wird die ROOT-Tabellen-Source erzeugt.

(KDCFILE,ROOTSRC)

Es wird die KCDFILE und die ROOT-Tabellen-Source erzeugt.

CLUSTER

Die Cluster-Konfigurationsdatei wird erzeugt.

Wenn Sie OPTION GEN=CLUSTER oder (CLUSTER,...) angegeben haben, müssen Sie auch eine CLUSTER-Anweisung und mindestens zwei CLUSTER-NODE-Anweisungen angeben.

(CLUSTER,KDCFILE)

Es werden die Cluster-Konfigurationsdatei, die KDCFILE und die Cluster-User-Datei erzeugt. Existiert bereits eine Cluster-User-Datei, dann wird diese ausgewertet und gegebenenfalls erweitert.

(CLUSTER,ROOTSRC)

Es werden die Cluster-Konfigurationsdatei und die ROOT-Tabellen-Source erzeugt.

(CLUSTER,KDCFILE,ROOTSRC)
Es werden die Cluster-Konfigurationsdatei, die KDCFILE, die Root-Tabellen-Source und die Cluster-User-Datei erzeugt. Existiert bereits eine Cluster-User-Datei, dann wird diese ausgewertet und gegebenenfalls erweitert

NO Es wird nur eine Überprüfung der Parameter durchgeführt.

ALL Die KDCFILE und die ROOT-Tabellen-Source werden erzeugt.

Wird eine Root-Tabellen-Source erzeugt, dann muss die ROOT-Anweisung angegeben werden. Dies ist der Fall bei folgenden Angaben:
- ROOTSRC
- ALL
- (CLUSTER,ROOTSRC)
- (CLUSTER,KDCFILE,ROOTSRC)

X/W Folgendes ist in Unix- und Windows-Systemen zu beachten:

X/W X/W X/W
- KDCDEF erzeugt die ROOT-Tabellen-Source als C/C++ Programm unter dem Namen *rootname*.c (siehe Anweisung ROOT) im Dateiverzeichnis *filebase* (siehe Anweisung MAX ...,KDCFILE=*filebase*...).

GEN-RSA-KEYS =
Gibt an, ob RSA-Schlüssel erzeugt werden sollen.

Dieser Parameter ist nur von Bedeutung, wenn für die Anwendung Objekte mit Verschlüsselung generiert werden, siehe Operand ENCRYPTION-LEVEL bei der TAC-, PTERM- und TPOOL-Anweisung.

YES Es werden RSA-Schlüssel erzeugt, falls in der Anwendung Objekte generiert werden, die eine Verschlüsselung erfordern.

Standard.

NO Es werden keine RSA-Schlüssel erzeugt.

GEN-RSA-KEYS=NO sollte nur dann verwendet werden, wenn nach dem KDCDEF-Lauf die benötigten RSA-Schlüssel per KDCUPD aus einer alten KDCFILE in die neue KDCFILE übertragen werden. Näheres zur Übertragung von RSA-Schlüsseln mit KDCUPD siehe Seite 585.

Sind in einer Anwendung Objekte mit Verschlüsselung generiert und sind keine RSA-Schlüssel verfügbar, dann kann die Anwendung nur mit Einschränkungen betrieben werden, d.h.:

- TACs mit Verschlüsselung können nicht aufgerufen werden,
- zu PTERMs oder TPOOLs, die mit Verschlüsselung generiert sind, kann keine Verbindung aufgebaut werden.

ROOTSRC=filename

Dieser Parameter ist nur dann von Bedeutung, wenn die ROOT-Tabellen-Source erzeugt wird.

filename kann maximal 54 Zeichen lang sein.

Es wird eine ROOT-Source mit dem CSECT-Namen *rootname* erzeugt und in der Datei *filename* abgelegt. *rootname* wird in der ROOT-Anweisung festgelegt.

Standard: ROOT.SRC.ASSEMB.*rootname*

OSI-CON - Logische Verbindungen zum OSI TP-Partner definieren

Mit der Steueranweisung OSI-CON ordnen Sie einem OSI-LPAP-Partner eine reale Partner-Anwendung zu, wenn die Kommunikation über das OSI TP-Protokoll abgewickelt werden soll. Mit der OSI-CON-Anweisung definieren Sie die logischen Verbindungen zwischen der lokalen UTM-Anwendung und einer Partner-Anwendung. Dazu geben Sie an:

- Den Namen des OSI TP-Zugriffspunktes (Access Point) in der lokalen Anwendung, über den die Verbindung aufgebaut werden soll. Den Zugriffspunkt definieren Sie mit einer ACCESS-POINT-Anweisung.
- Die Adresse des OSI TP-Zugriffspunktes (Access Point) der Partner-Anwendung. Die Adresse besteht aus P-Selektor, S-Selektor, N-Selektor und T-Selektor.

X/W X/W Unter Unix- und Windows-Systemen dienen die folgenden Operanden zur Beschreibung des T-Selektors:

X/W X/W X/W X/W X/W
- TRANSPORT-SELECTOR (=Adresse der Partner-Anwendung im Partner-Rechner)
- T-PROT (das verwendete Transportprotokoll)
- TSEL-FORMAT (Formatindikator des T-Selektors)
- LISTENER-PORT (Portnummer für RFC1006)

X/W X/W Siehe dazu Abschnitt „Adressinformationen für das Transportsystem CMX bereitstellen (Unix- und Windows-Systeme)“ auf Seite 105.

Die Partner-Anwendung baut die Verbindung zur lokalen Anwendung über einen OSI-LPAP-Partner auf, den Sie in der OSI-LPAP-Anweisung beschreiben. Hier generieren Sie die Anzahl der Verbindungen und die Namen der einzelnen Verbindungen etc. Die Kommunikationsparameter des OSI-LPAP-Partners ordnen Sie der OSI-CON-Anweisung über den Parameter OSI-LPAP=*osi_lpap_name* zu. Die logische Verbindung wird so mit nur einer OSI-CON-Anweisung generiert, auch wenn zur Partner-Anwendung parallele Verbindungen existieren.

Ist eine Partner-Anwendung zu verschiedenen Zeiten in verschiedenen fernen Systemen erreichbar, so können Sie dem OSI-LPAP-Partner, dem diese Partner-Anwendung zugeordnet ist, mehrere Adressen zuordnen und damit Ersatzverbindungen definieren. Für die Generierung weisen Sie einer OSI-LPAP-Anweisung (siehe Seite 429) mehrere OSI-CON-Anweisungen zu (OSI-CON-Anweisungen mit demselben *osi_lpap_name* und demselben LOCAL-ACCESS-POINT). Es darf jedoch nur eine OSI-CON-Anweisung aktiv gesetzt werden. Umschalten auf eine Ersatzverbindung können Sie dann per Administration.

Wenn Sie OSI-LPAP-Bündel verwenden, so gilt für die OSI-CONs der Slave-LPAPs eines LPAP-Bündels außerdem:
Alle OSI-CONs aller Slave-LPAPs eines LPAP-Bündels müssen dem gleichen Access Point zugeordnet sein (siehe auch Abschnitt „MASTER-OSI-LPAP - Master-LPAP eines OSI-LPAP-Bündels definieren“ auf Seite 367“).

```
OSI-CON␣      connection_name
              [ ,ACTIVE={ YES | NO } ]
                ,LOCAL-ACCESS-POINT=access_point_name
                ,NETWORK-SELECTOR=C'c'
                ,OSI-LPAP=osi_lpap_name
                ,PRESENTATION-SELECTOR={ *NONE |
                                         (C'c' [, STD | EBCDIC | ASCII ]) |
                                         X'xx' }
                ,SESSION-SELECTOR={ *NONE |
                                    (C'c' [, STD | EBCDIC | ASCII ]) |
                                    X'xx' }
                ,TRANSPORT-SELECTOR=C'c'
```

X/W ***zusätzliche Operanden in Unix-, Windows-Systemen***

```
X/W           [ ,LISTENER-PORT=number ]
X/W           [ ,MAP={ USER | SYSTEM } ]
X/W           [ ,T-PROT=RFC1006 ]
X/W           [ ,TSEL-FORMAT={ T | E | A } ]
```

connection_name

Name der logischen Verbindung zwischen der lokalen UTM-Anwendung und der Partner-Anwendung für die Kommunikation über das OSI TP-Protokoll. *connection_name* identifiziert die Verbindung in der lokalen Anwendung. *connection_name* kann max. 8 Zeichen lang sein und muss in der lokalen Anwendung eindeutig sein.

ACTIVE= Logische Verbindung zur Partner-Anwendung aktiv oder inaktiv setzen. Für Ersatzverbindungen zur Partner-Anwendung werden mehrere OSI-CON-Steueranweisungen mit demselben *osi_lpap_name* eines OSI-LPAP-Partners generiert. Aber nur eine OSI-CON-Anweisung darf mit ACTIVE=YES generiert werden, alle weiteren müssen mit ACTIVE=NO generiert werden. Auf die Ersatzverbindungen kann per Administration umgeschaltet werden.

YES Die mit dieser OSI-CON-Anweisung definierte Verbindung aktiv setzen.

Standard: YES

NO Die mit dieser OSI-CON-Anweisung definierte Verbindung inaktiv setzen.

X/W X/W X/W LISTENER-PORT=number

Portnummer der Partner-Anwendung, wenn die Verbindung zur Partner-Anwendung über TCP/IP aufgebaut wird.

X/W Erlaubte Werte: 102 und 1025 - 32767

X/W X/W X/W Standard: 0 (keine Portnummer)
Bei OPTION CHECK-RFC1006=YES muss bei LISTENER-PORT eine Portnummer angegeben werden.

LOCAL-ACCESS-POINT=access_point_name

Name des lokalen OSI TP-Zugriffspunktes (Access Point), über den die Kommunikation mit der Partner-Anwendung erfolgen soll. Der Zugriffspunkt wird mit einer ACCESS-POINT-Steueranweisung definiert.

Wenn für den OSI-LPAP-Partner, dem die Partner-Anwendung zugeordnet ist, Ersatzverbindungen definiert werden (mehrere OSI-CON-Anweisungen mit demselben *osi_lpap_namen*) muss in allen Ersatzverbindungen immer derselbe LOCAL-ACCESS-POINT angegeben werden.

Wenn der Application Context des OSI-LPAP-Partners die CCR-Syntax verwendet, müssen für den LOCAL-ACCESS-POINT die folgenden Adresskomponenten ebenfalls definiert sein:

- APPLICATION-ENTITY-QUALIFIER (siehe ACCESS-POINT-Anweisung auf Seite 275)
- APPLICATION-PROCESS-TITLE (siehe UTMD-Anweisung auf Seite 553)

X/W X/W X/W X/W MAP= regelt die Umwandlung ASCII/EBCDIC beim Austausch unformatierter Nachrichten mit Partner-Anwendungen.
Bei formatierten Nachrichten (KCMF enthält ein Formatkennzeichen) führt openUTM grundsätzlich keine Nachrichtenbehandlung durch.

X/W X/W X/W USER openUTM führt keine Nachrichtenbehandlung durch, d.h. die Daten des Nachrichtenbereichs werden unverändert an die Partner-Anwendung übertragen.

X/W Standard: USER

X/W X/W X/W X/W SYSTEM openUTM konvertiert die Daten des Nachrichtenbereichs vor dem Absenden von ASCII nach EBCDIC bzw. nach dem Empfangen von EBCDIC nach ASCII. Die Nachricht darf dabei nur abdruckbare Zeichen enthalten und muss im Zeilenmodus (KCMF = Leerzeichen) erzeugt werden.

NETWORK-**SEL**ECTOR=C’c’

Name des Partner-Rechners. Der Name darf max. 8 Zeichen lang sein.

Die Angabe von N-SEL ist Pflicht.

B
B
B

BS2000/OSD:
In N-SEL= müssen Sie den BCAM-Prozessornamen des Rechners angeben, auf dem sich die Partner-Anwendung befindet.

X/W
X/W

Unix-, Windows-Systeme:
Für die Angabe von N-SEL haben Sie zwei Möglichkeiten:

X/W
X/W
X/W
X/W

– Sie geben den realen Hostnamen an, unter dem die IP-Adresse des Partner-Rechners im Name Service des lokalen Systems (z.B. der hosts-Datei) eingetragen ist. Einen Aliasnamen des Rechners dürfen Sie nicht angeben.

X/W
X/W
X/W
X/W
X/W

– Sie geben den UTM-Hostnamen des Partner-Rechners an. Voraussetzung hierfür ist, dass Sie die Umgebungsvariable UTM_NET_HOSTNAME gesetzt und den UTM-Hostnamen in der Umwandlungsdatei festgelegt haben (siehe Abschnitt „Mapped Hostnamen verwenden (Unix- und Windows-Systeme)“ auf Seite 118).

Bitte beachten Sie, dass das Namenspaar (TRANSPORT-SELECTOR, NETWORK-SELECTOR) weder mit dem Namenspaar (*remote_appliname*, PRONAM) in einer CON-Anweisung (siehe Seite 309) noch mit dem Namenspaar (*ptermname*, PRONAM) in einer PTERM-Anweisung (siehe Seite 440) übereinstimmen darf.

OSI-LPAP=osi_lpap_name

Name des OSI-LPAP-Partners, der in der lokalen Anwendung als logischer Anschlusspunkt für die Partner-Anwendung definiert wurde.
osi_lpap_name muss mit einer Anweisung OSI-LPAP definiert sein.
osi_lpap_name kann max. 8 Zeichen lang sein.

PRESENTATION-**SEL**ECTOR=

Presentation-Selektor der Partner-Anwendung; Adresskomponente des OSI-TP-Zugriffspunktes im System des fernen Partners. Der angegebene Wert muss mit dem in der Partner-Anwendung angegebenen Presentation-Selektor für diesen Zugriffspunkt übereinstimmen.

*NONE — Es wird kein symbolischer Presentation-Selektor vergeben.

C’c’ — Der Presentation-Selektor wird als Characterstring (c) angegeben. Der für *c* angegeben Wert darf maximal 16 Zeichen lang sein. Klein- und Großbuchstaben werden unterschieden.

Bei einem Characterstring können Sie wählen, in welchem Code die Zeichen interpretiert werden:

STD Die Zeichen werden in Maschinen-spezifischer Codierung interpretiert (BS2000/OSD = EBCDIC; Unix- und Windows-Systeme = ASCII).

Standard: STD

EBCDIC
Die Zeichen werden in EBCDIC-Codierung interpretiert.

ASCII Die Zeichen werden in ASCII-Codierung interpretiert.

X'x' Der Presentation-Selektor wird als Hexadezimalzahl (x) angegeben. Für x dürfen maximal 32 Hexadezimalziffern (≙16 Byte) angegeben werden. Es muss eine gerade Anzahl von Hexadezimalziffern angegeben werden.

SESSION**-SEL**ECTOR=
Session-Selektor der Partner-Anwendung; Adresskomponente des OSI TP-Zugriffspunktes im System des fernen Partners. Der angegebene Wert muss mit dem in der Partner-Anwendung angegebenen Session-Selektor für diesen Zugriffspunkt übereinstimmen.

*NONE Es soll kein Session-Selektor vergeben werden.

C'c' Der Session-Selektor wird als Characterstring (c) angegeben. Der für c angegebene Wert darf maximal 16 Zeichen lang sein. Klein- und Großbuchstaben werden unterschieden.

Bei einem Characterstring können Sie wählen, in welchem Code die Zeichen interpretiert werden:

STD Die Zeichen werden in Maschinen-spezifischer Codierung interpretiert (BS2000/OSD=EBCDIC; Unix- und Windows-Systeme=ASCII).

Standard: STD

EBCDIC
Die Zeichen werden in EBCDIC-Codierung interpretiert.

ASCII Die Zeichen werden in ASCII-Codierung interpretiert.

X'x' Der Session-Selektor wird als Hexadezimalzahl (x) angegeben. Für x dürfen maximal 32 Hexadezimalziffern (≙16 Byte) angegeben werden. Es muss eine gerade Anzahl von Hexadezimalziffern angegeben werden.

TRANSPORT**-SEL**ECTOR=C'c'
Transport-Selektor der Partner-Anwendung

Für den Transport-Selektor können Sie max. 8 abdruckbare Zeichen angeben. Zulässige Zeichen sind Großbuchstaben, Ziffern und die Zeichen $, # und @. Bindestriche im Namen sind nicht erlaubt. Das erste Zeichen muss ein Großbuchstabe sein.

Die Angabe von T-SEL=C'c' ist Pflicht.

Folgendes müssen Sie für T-SEL= angeben:

B
B
– *BS2000/OSD:*
BCAM-Anwendungsname des fernen Partners

X/W
X/W
– *Unix-, Windows-Systeme:*
T-Selektor der Partner-Anwendung.

X/W
X/W
X/W
Bei OPTION CHECK-RFC1006=YES müssen Sie in T-SEL den T-Selektor angeben, der der Partner-Anwendung im fernen System zugeordnet ist.

Bitte beachten Sie, dass das Namenspaar (TRANSPORT-SELECTOR, NETWORK-SELECTOR) weder mit dem Namenspaar (*remote_appliname*, PRONAM) in einer CON-Anweisung (auf Seite 309) noch mit dem Namenspaar (*ptermname*, PRONAM) in einer PTERM-Anweisung (auf Seite 440) übereinstimmen darf.

X/W
X/W
T-PROT= Adressformat, mit dem sich der OSI TP-Partner beim Transportsystem anmeldet.

X/W
X/W
Informationen zu den folgenden Adressformaten finden Sie im CMX-Benutzerhandbuch oder in der Online-Hilfe zu PCMX.

X/W
X/W
RFC1006 Adressformat RFC1006
ISO-Transportprotokoll über TCP/IP und Konvergenzprotokoll RFC1006.

X/W
Standard: RFC1006

X/W
X/W
X/W
X/W
X/W
TSEL-FORMAT=
Formatindikator des T-Selektors
Der Formatindikator gibt die Codierung des T-Selektors im Transportprotokoll an. Nähere Informationen finden Sie im CMX-Benutzerhandbuch bzw. in der Online-Hilfe zu PCMX.

X/W
T TRANSDATA-Format

X/W
E EBCDIC-Zeichenformat

X/W
A ASCII-Zeichenformat

X/W
X/W
X/W
X/W
Standard:
T wenn der Zeichenvorrat des Wertes von T-SEL dem TRANSDATA-Format entspricht
E sonst

X/W
X/W
Für den Betrieb über TCP/IP mit RFC1006 wird jedoch empfohlen, explizit einen Wert für TSEL-FORMAT anzugeben.

OSI-LPAP - OSI-LPAP-Partner für verteilte Verarbeitung über OSI TP definieren

Mit der Steueranweisung OSI-LPAP definieren Sie in der lokalen Anwendung einen logischen Anschlusspunkt für eine Partner-Anwendung, mit der Sie über das OSI TP-Protokoll kommunizieren wollen. Der logische Anschlusspunkt für Partner-Anwendungen heißt OSI-LPAP-Partner. Für ihn legen Sie einen logischen Partnernamen und die folgenden logischen Verbindungseigenschaften fest:

- Sie geben den Application Entity Title (AET) der Partner-Anwendung an. Der AET wird dann benötigt, wenn mit Transaktionssicherung (Commit Functional Unit) gearbeitet wird, oder aber ein heterogener Partner einen AET für den Verbindungsaufbau erwartet. Der AET setzt sich aus den folgenden Komponenten zusammen, die für die Partner-Anwendung definiert und hier angegeben werden:
  - Application Entity Qualifier (AEQ) des fernen Zugriffspunktes (siehe ACCESS-POINT-Anweisung auf Seite 275)
  - Application Process Title (APT) der Partner-Anwendung (siehe UTMD-Anweisung auf Seite 553)
- Sie müssen den Application Context angeben, der bei der Kommunikation über das OSI TP-Protokoll mit der Partner-Anwendung verwendet wird. Wenn Sie keinen Standard-Application Context verwenden, definieren Sie Ihren Application Context mit der APPLICATION-CONTEXT-Anweisung, siehe Seite 283. Wenn der Application Context des OSI-LPAP-Partners die CCR-Syntax enthält, müssen für die Partner-Anwendung ein AEQ und ein APT definiert werden.
- Anzahl und Eigenschaften der Verbindungen zur Partner-Anwendung.
- Zugriffsrechte der Partner-Anwendung in der lokalen Anwendung.
  Zur Definition der Zugriffsrechte stehen die Operanden KSET und ASS-KSET zur Verfügung. In KSET legen Sie die maximalen Zugriffsrechte des OSI TP-Partners fest, die der OSI TP-Partner hat, wenn er sich mit einer Benutzerkennung bei der lokalen Anwendung anmeldet. Mit dem Operanden ASS-KSET können Sie diese Zugriffsrechte einschränken. Die eingeschränkten Zugriffsrechte werden wirksam, wenn der OSI TP-Partner bei der Anmeldung keine Benutzerkennung übergibt, d.h. der „Association-User“ aktiv ist.
- Administrationsberechtigung der Partner-Anwendung in der lokalen Anwendung.
- Maximalwerte für die Message Queue des OSI-LPAP-Partners.

Ist ein Kommunikationspartner zu verschiedenen Zeiten in verschiedenen fernen Systemen erreichbar, so können Sie diesem Partner mehrere Adressen zuordnen. Dazu weisen Sie einer OSI-LPAP-Anweisung mehrere OSI-CON-Anweisungen zu (OSI-CON-Anweisungen mit demselben *osi_lpap_name,* siehe Seite 423). Es darf jedoch nur eine OSI-CON-Anweisung aktiv gesetzt werden. Umschalten auf eine Ersatzverbindung können Sie dann per Administration. Alle zu einem OSI-LPAP-Partner gehörenden OSI-CON-Verbindungen müssen denselben LOCAL-ACCESS-POINT haben.

Es können maximal 21000 Associations generiert werden.

```
OSI-LPAP␣     osi_lpap_name
                ,APPLICATION-CONTEXT=application_context_name
              [ ,APPLICATION-ENTITY-QUALIFIER=application_entity_qualifier
                ,APPLICATION-PROCESS-TITLE=object_identifier ]
              [ ,ASS-KSET=keysetname2 ]
                ,ASSOCIATION-NAMES=association_name
              [ ,ASSOCIATIONS=number ]
              [ ,BUNDLE=master-lpap-name]
              [ ,CONNECT=number ]
                ,CONTWIN=number
              [ ,IDLETIME=time ]
              [ ,KSET=keysetname1 ]
              [ ,PERMIT={ ADMIN | SATADM[1] | ( ADMIN,SATADM )[1] } ]
              [ ,QLEV=number ]
              [ ,STATUS={ ON | OFF } ]
              [ ,TERMN=termn_id ]
```

B [1] Nur in BS2000/OSD erlaubt

osi_lpap_name

Name des OSI-LPAP-Partners der Partner-Anwendung, über den die Teilprogramme der lokalen UTM-Anwendung die Partner-Anwendung ansprechen. *osi_lpap_name* kann max. 8 Zeichen lang sein.

osi_lpap_name muss eindeutig sein und darf keinem weiteren Objekt der Namensklasse 1 zugeordnet sein. Siehe dazu auch Abschnitt „Eindeutigkeit der Namen und Adressen“ auf Seite 262.

APPLICATION-CONTEXT=application_context_name

Name des Application Context, der von der Partner-Anwendung verwendet werden soll.

Pflichtoperand

openUTM unterstützt standardmäßig die folgenden Application Contexts:

- UDTAC
- UDTDISAC
- XATMIAC
- UDTCCR
- UDTSEC
- XATMICCR

Siehe dazu auch die APPLICATION-CONTEXT-Anweisung auf Seite 283. Wenn der generierte Application Context nicht mit dem von der Partner-Anwendung verwendeten übereinstimmt, lehnt openUTM den Verbindungsaufbau ab mit folgenden UTM-Meldungen:

```
P001 APPLICATION CONTEXT NOT SUPPORTED    oder
P011 Abstract Syntax nicht erlaubt
```

APPLICATION-**E**NTITY-**Q**UALIFIER=application_entitiy_qualifier

Application-Entity-Qualifier der Partner-Anwendung. Der Application-Entity-Qualifier wird zusammen mit dem Application-Process-Title zur Adressierung einer Partner-Anwendung bei heterogener Kopplung verwendet bzw. wenn mit Transaktionssicherung (Commit Functional Unit) gearbeitet wird. *application_entitiy_qualifier* ist eine positive ganze Zahl, über die die Partner-Anwendung im fernen System gerufen wird.

Enthält der Application Context die CCR-Syntax, dann ist dieser Parameter Pflicht. Das Namenspaar *application_entitiy_qualifier* und *object_identifier* muss innerhalb der UTM-Anwendung eindeutig sein.

Der *application_entitiy_qualifier*, den Sie hier angeben, muss in der Partner-Anwendung einem Zugriffspunkt zugeordnet sein.

Minimalwert: 1
Maximalwert: 67108 863 (2^{26}-1)

APPLICATION-**P**ROCESS-**T**ITLE=object_identifier

Für *object_identifier* ist der Application Process Title der Partner-Anwendung anzugeben. Der Application Process Title wird zusammen mit dem Application Entity Qualifier zur Adressierung einer Partner-Anwendung bei heterogener Kopplung verwendet bzw. wenn mit Transaktionssicherung (Commit Functional Unit) gearbeitet wird.

Enthält der Application Context die CCR-Syntax, dann ist dieser Parameter Pflicht. Das Namenspaar *application_entitiy_qualifier* (der OSI-LPAP-Anweisung) und *object_identifier* muss innerhalb der UTM-Anwendung eindeutig sein.

Zur Definition des APT siehe die UTMD-Anweisung auf Seite 553.

ASS-KSET=keysetname2

ist nur erlaubt, wenn die lokale Anwendung mit Benutzerkennungen generiert wird. ASS-KSET dürfen Sie nur zusammen mit KSET setzen.

Für *keysetname2* ist der Name eines Keyset anzugeben. Das Keyset muss mit einer KSET-Anweisung definiert werden.

Mit ASS-KSET= legen Sie die minimalen Zugriffsrechte fest, die die Partner-Anwendung in der lokalen Anwendung ausüben kann.

Das in *keysetname2* angegebene Keyset wird dann wirksam, wenn die Partner-Anwendung beim Aufbau der Association keine Benutzerkennung an openUTM übergibt.
Die Zugriffsrechte ergeben sich dann aus der Menge der Keycodes, die sowohl in dem mit KSET= als auch in dem mit ASS-KSET= generierten Keyset enthalten sind (Schnittmenge). Aus diesem Grund sollten alle Keycodes, die ASS-KSET=*keysetname2* enthalten, auch in KSET=*keysetname1* enthalten sein.

Standard: kein Keyset
Es gelten immer die in KSET festgelegten Zugriffsrechte

ASSOCIATION**-NAMES**=association_name

Name, der für die logischen Verbindungen zur Partner-Anwendung innerhalb der lokalen Anwendung vergeben wird.

Der Name der Verbindungen setzt sich zusammen aus dem Wert von *association_name* als Präfix und einer Laufnummer. Die Laufnummer liegt zwischen 1 und dem Wert des Operanden ASSOCIATIONS, d.h. der Anzahl der parallelen Verbindungen. Der gesamte Name für eine Verbindung darf max. 8 Zeichen lang sein. Die maximale Länge des für *association_name* angegebenen Wertes ist abhängig von dem bei ASSOCIATIONS angegebenen Wert. Für die Anzahl der Verbindungen gilt:

Anzahl der Dezimalstellen des Wertes von ASSOCIATIONS +
Anzahl der Zeichen von *association_name* ≤ 8

Beispiel

Angegeben wird: ASSOCIATIONS=10,
ASSOCIATION-NAMES=ASSOC

Namen der Verbindungen: ASSOC01, ASSOC02, ..., ASSOC10

Diese Namen werden in der lokalen UTM-Anwendung als Associationnamen benutzt. Sie dürfen keine Benutzerkennung (USER) und keine Sessionnamen für verteilte Verarbeitung über LU6.1 (LSES) mit demselben Namen generieren.

ASSOCIATIONS=number

Maximale Anzahl der parallelen Verbindungen zur Partner-Anwendung. Die maximale Anzahl paralleler Verbindungen zu einer Partner-Anwendung ist abhängig von den Schichten 1 - 6 des OSI-Referenzmodells von ISO (insbesondere der Transportschicht, Schicht 4).

Die Anzahl der parallelen Verbindungen muss mit der Generierung der Partner-Anwendung abgestimmt werden.

Standard: 1
Minimalwert: 1
Maximalwert: Die maximale Anzahl der Associations wird durch die Größe des Namensraums der UTM-Anwendung begrenzt (siehe Abschnitt „Anzahl der Namen“ auf Seite 259).

BUNDLE=master-lpap-name

Name eines Master-LPAP. Durch die Angabe von *master-lpap-name* wird dieser OSI-LPAP-Partner zu einem Slave-LPAP des zugehörigen Master-LPAP.

Das Master-LPAP, das hier angegeben wird, muss mit einer Anweisung MASTER-OSI-LPAP definiert werden.

CONNECT=number

Anzahl der Verbindungen, die beim Start der lokalen Anwendung zu dieser Partner-Anwendung automatisch aufgebaut werden sollen. Der Aufbau einer Verbindung beim Start der Anwendung kann sowohl in der lokalen Anwendung als auch bei der Partner-Anwendung angefordert werden. Dadurch erreicht man, dass die Verbindung automatisch aufgebaut wird, sobald beide Partner verfügbar sind.

Standard: 0
Maximalwert: Anzahl der parallelen Verbindungen, die beim Operanden ASSOCIATIONS= angegeben wurde.

CONTWIN=number

Anzahl der Verbindungen, für die die lokale Anwendung der Contention Winner sein soll. Für die restlichen Verbindungen (Angabe bei ASSOCIATIONS minus Angabe bei CONTWIN) ist die lokale Anwendung der Contention Loser.

Der Contention Winner einer Verbindung übernimmt die Verwaltung der Verbindung. Aufträge können aber sowohl vom Contention Winner als auch vom Contention Loser gestartet werden. Im Konfliktfall, wenn beide Kommunikationspartner gleichzeitig einen Auftrag starten wollen, wird die Verbindung vom Auftrag des Contention Winner belegt.

Pflichtoperand

Die Anzahl der Contention Winner und Contention Loser muss mit der Generierung der Partner-Anwendung abgestimmt werden.
Contention Winner sollte für die Anwendung generiert werden, die am häufigsten Aufträge startet.

Minimalwert: 0
Maximalwert: Anzahl der parallelen Verbindungen, die beim Operanden ASSOCIATIONS= angegeben wurde.

IDLETIME=time

Zeit in Sekunden zur Überwachung des Idle-Zustands einer Verbindung. Wird während der mit *time* festgelegten Zeit die Verbindung von keinem Auftrag belegt, baut openUTM die Verbindung ab.

IDLETIME=0 bewirkt, dass der Idle-Zustand nicht überwacht wird.

Standard: 0
Minimalwert: 0
Maximalwert: 32767

KSET=keysetname1

legt die maximalen Zugriffrechte der Partner-Anwendung in der lokalen Anwendung fest. In *keysetname1* ist der Name eines Keysets anzugeben. Das Keyset muss mit einer KSET-Anweisung definiert werden.

Übergibt der OSI TP-Partner für einen OSI TP-Dialog keine Benutzerkennung an die lokale Anwendung, dann ergeben sich seine Zugriffsrechte für diesen OSI TP-Dialog aus der Menge der Keycodes, die sowohl in dem mit KSET= als auch in dem mit ASS-KSET= generierten Keyset vorhanden sind (Schnittmenge).
Das Keyset *keysetname1* sollte deshalb alle Keycodes enthalten, die auch in dem mit ASS-KSET= generierten Keyset enthalten sind.

Übergibt der OSI TP-Partner eine Benutzerkennung, dann ergeben sich die Zugriffsrechte für diesen OSI TP-Dialog aus der Menge der Keycodes, die sowohl in dem Keyset der Benutzerkennung als auch in dem mit KSET generierten Keyset des OSI-LPAP enthalten sind.

Standard: kein Keyset,
d.h. es dürfen nur Services gestartet bzw. in der lokalen Anwendung generierte, entfernte Services (LTAC) adressiert werden, die nicht mit Lockcodes gesichert sind.

PERMIT= legt die Berechtigungsstufe der Partner-Anwendung fest.

ADMIN Die Partner-Anwendung darf Administrationsfunktionen in der lokalen Anwendung ausführen.

B B B B SATADM Die Partner-Anwendung kann SAT-Preselection-Funktionen in der lokalen Anwendung ausführen. D.h. die Partner-Anwendung kann die SAT-Protokollierung bestimmter Ereignisse ein- bzw. ausschalten (UTM-SAT-Administrations-Berechtigung).

B B B (ADMIN,SATADM)
Die Partner-Anwendung kann Administrations- und SAT-Preselection-Funktionen in der lokalen Anwendung ausführen.

B B B B Standard:
Wenn der Operand nicht angegeben wird, kann die Partner-Anwendung keine Administrations- und SAT-Preselection-Funktionen in der lokalen Anwendung ausführen.

QLEV=queue_level_number
gibt die maximale Anzahl der asynchronen Nachrichten an, die in der Message Queue des OSI-LPAP-Partners stehen dürfen. Wird der Schwellwert überschritten, so werden weitere APRO-AM-Aufrufe an diesen LPAP-Partner mit der Meldung 40Z abgewiesen.

Standard: 32767
Minimalwert: 0
Maximalwert: 32767 (d.h., keine Beschränkung der Queue-Länge)

STATUS= legt fest, ob der OSI-LPAP-Partner gesperrt ist. Der Status kann im Betrieb mit dem Administrationskommando KDCLPAP geändert werden.

ON Der OSI-LPAP-Partner ist nicht gesperrt. Es können Verbindungen zwischen der Partner-Anwendung und der lokalen Anwendung aufgebaut werden bzw. es bestehen bereits Verbindungen.

Standard: ON

OFF Der OSI-LPAP-Partner ist gesperrt. Es können keine Verbindungen zwischen der Partner-Anwendung und der lokalen Anwendung aufgebaut werden.

TERMN=termn_id
Max. 2 Zeichen langes Kennzeichen für die Art des Kommunikationspartners. *termn_id* wird nicht von openUTM abgefragt, sondern vom Benutzer zur Auswertung gesetzt, um beispielsweise Terminaltypen abzufragen oder zu gruppieren etc. *termn_id* wird im KB-Kopf der Vorgänge eingetragen, die von der Partner-Anwendung in der lokalen Anwendung gestartet wurden.

Standard: A6

PROGRAM - Teilprogramme definieren

Mit der PROGRAM-Anweisung werden die Namen und Eigenschaften der Teilprogramme vereinbart.
Soll im KDCDEF-Lauf eine ROOT-Tabellen-Source erzeugt werden (Anweisung OPTION mit GEN=ROOTSRC oder GEN=ALL), dann müssen Sie mindestens eine PROGRAM-Anweisung angeben.

B **UTM-Teilprogramme in BS2000/OSD generieren**

```
PROGRAM␣      objectname
              , COMP={ ASSEMB |
                       C |
                       COB1 |
                       FOR1 |
                       PASCAL-XT|
                       PLI1 |
                       SPL4 |
                       ILCS }
              [ ,LOAD-MODULE=lmodname ]
```

objectname bezeichnet den Einsprungpunkt für ein Teilprogramm (CSECT- oder ENTRY-Name). *objectname* darf maximal 32 Zeichen lang sein.

Details zu den erlaubten Zeichen siehe Abschnitt „Format der Namen“ auf Seite 257.

COMP= bezeichnet das Laufzeitsystem, das für das Teilprogramm verwendet werden soll.

Pflichtoperand

Für alle Teilprogramme, die ILCS (Inter Language Communication Services) unterstützen, müssen Sie COMP=ILCS angeben (z.B. Teilprogramme in COBOL85, FORTRAN90, C etc.).
Ob ein Teilprogramm ILCS unterstützt, hängt ab von der beim Übersetzen eingesetzten Compilerversion und von der Version des Laufzeitsystems, unter der das Teilprogramm abläuft.

Weitere Hinweise zur Wahl des Parameters COMP finden Sie im Anhang vom openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“. Diese sollten Sie insbesondere dann beachten, wenn Ihre Programme mit älteren Compiler-Versionen übersetzt wurden!

B

B
B
B
B

COMP=C ist synonym für COMP=ILCS

Das Administrationsprogramm KDCADM muss mit COMP=ILCS generiert werden und mit einer TAC-Anweisung muss zumindest der Transaktionscode KDCSHUT zugeordnet werden.

B
B
B
B
B
B
B
B
B

LOAD-MODULE=lmodname

LOAD-MODULE bezeichnet den Namen des Lademoduls, in dem das Teilprogramm gebunden wurde. Das Lademodul muss mit einer Steueranweisung LOAD-MODULE definiert werden. *lmodname* kann max. 32 Zeichen lang sein.
Für die Namensvergabe gelten die gleichen Regeln wie für Elementnamen einer Programm-Bibliothek (siehe auch Abschnitt „Format der Namen“ auf Seite 257).

B
B
B
B
B

Wenn Sie den Operanden LOAD-MODULE verwenden, müssen Sie Folgendes beachten:

– Das Administrationsprogramm KDCADM darf keinem Lademodul zugeordnet werden, das mit LOAD-MODE=ONCALL in der LOAD-MODULE-Anweisung generiert ist.

X/W **UTM-Teilprogramme in Unix- bzw. Windows-Systemen generieren**

```
X/W  PROGRAM␣      objectname
X/W                [ ,COMP={ C | COB2 | CPP } ]
X/W                [ ,SHARED-OBJECT=shared_object_name ]
```

X/W objectname — Name des Einsprungpunktes des Teilprogramms. Der Name darf max. 32 Zeichen lang sein. Details zu den erlaubten Zeichen finden Sie in Abschnitt „Format der Namen“ auf Seite 257.

X/W COMP= — bezeichnet den Compiler, mit dem das Teilprogramm übersetzt wurde.

X/W C — C-Compiler

X/W Standard: C

X/W CPP — C++-Compiler

X/W COB2 — COBOL-Compiler (Server Express / NetExpress)
X In Unix-Systemen wird ein Microfocus COBOL-Programm generiert, das mit Server Express von MicroFocus erstellt wurde.
X Verwenden Sie nur Ziffern und Großbuchstaben für die PROGRAM-ID und die Einsprungpunkte (*objectname*). Das entspricht den IBM-Konventionen und garantiert die Portabilität der Programme.

W In Windows-Systemen wird ein COBOL-Programm generiert, das mit NetExpress von MicroFocus erstellt wurde.

X/W SHARED-OBJECT=shared_object_name
(Programmaustausch mit Hilfe des dynamischen Binders)
Der Operand muss nur angegeben werden, wenn das Teilprogramm nachgeladen werden soll. *shared_object_name* ist der Name des Shared Objects (Unix-System) bzw. der DLL (Windows-System), in das das Teilprogramm eingebunden wurde. Dieses Shared Object/DLL muss mit einer SHARED-OBJECT-Anweisung definiert sein.

PTERM - Eigenschaften von Clients und Druckern und Zuordnung zum LTERM-Partner festlegen

Mit der PTERM-Anweisung definieren Sie die Eigenschaften der physikalischen Clients und Drucker der UTM-Anwendung.

Clients sind Terminals, UPIC-Clients und Transportsystem-Anwendungen. Unter Transportsystem-Anwendungen (kurz TS-Anwendungen) versteht man alle Anwendungen, die als PTYPE=APPLI oder PTYPE=SOCKET generiert sind. Siehe dazu auch die Tabelle beim Operanden PTYPE auf Seite 454 (BS2000/OSD) bzw. Seite 456 (Unix-, Windows-Systeme).

Für Clients müssen Sie immer dann eine PTERM-Anweisung absetzen, wenn von der lokalen UTM-Anwendung aus eine Verbindung zum Client/Drucker aufgebaut werden soll.

Mit der PTERM-Anweisung ordnen Sie einem Client/Drucker einen LTERM-Partnern zu. Der LTERM-Partner wird in der LTERM-Anweisung definiert. Für jeden Client/Drucker ist dabei eine LTERM-Anweisung zu schreiben, siehe auch Beschreibung der LTERM-Anweisung auf Seite 354.

Sie können aber auch zunächst Clients/Drucker in PTERM-Anweisungen beschreiben und die Zuordnung zu LTERM-Partnern später im Betrieb per Administration vornehmen. Ausnahmen sind UPIC-Clients und TS-Anwendungen, ihnen müssen Sie sofort einen LTERM-Partner zuordnen.

Wenn keine LTERM-Pools (TPOOL-Anweisung, siehe Seite 518) generiert sind, müssen Sie in mindestens einer PTERM-Anweisung im Operanden LTERM= einen Client zuweisen. Nur dann kann eine Verbindung zur Anwendung aufgebaut werden, über die Sie mit der Anwendung arbeiten können.

W

Von openUTM auf Windows-Systemen werden Drucker nicht unterstützt.

Adresse des Client/Druckers

Damit die Anwendung Verbindungen zum Client/Drucker aufbauen kann, müssen Sie seine Adresse angeben. Dazu dienen die Operanden

- *ptermname* (Name/T-Selektor des Kommunikationspartners)
- PRONAM (Name des Partner-Rechners) [1)]
- LISTENER-PORT (TCP/IP-Portnummer) [2)]

X/W
X/W
X/W

1) Unter Unix- und Windows-Systemen darf der Name des Partner-Rechners nur angegeben werden, wenn der Partner ein remote UPIC-Client (UPIC-R) oder eine TS-Anwendung (PTYPE= APPLI oder SOCKET) ist.

B
B

2) Unter BS2000/OSD darf LISTENER-PORT nur bei PTYPE=SOCKET angegeben werden.

X/W X/W Zur weiteren Definition der Partneradresse dienen unter Unix- und Windows-Systemen folgende Operanden:

X/W – T-PROT (Adressformat für das verwendete Transportprotokoll)
X/W – TSEL-FORMAT (Formatindikator des T-Selektors)

X/W X/W X/W Siehe dazu Abschnitt „Adressinformationen für das Transportsystem CMX bereitstellen (Unix- und Windows-Systeme)“ auf Seite 105 bzw. Abschnitt „Adressinformationen für das Transportsystem SOCKET bereitstellen (Unix- und Windows-Systeme)“ auf Seite 114.

Wird die Verbindung zu einer TS-Anwendung über die Socket-Schnittstelle mit native TCP/IP als Transportprotokoll aufgebaut, dann müssen Sie in PRONAM den Rechner angeben, an dem die TS-Anwendung abläuft, und in LISTENER-PORT die Portnummer innerhalb des Partner-Rechners, an der die TS-Anwendung auf Verbindungsaufbauwünsche aus dem Netz wartet. In BCAMAPPL müssen Sie einen Anwendungsnamen angeben, für den T-PROT=SOCKET generiert wurde (siehe auch BCAMAPPL-Anweisung auf Seite 289).
Mit dem Operanden MAP können Sie festlegen, ob openUTM eine Nachrichtenbehandlung durchführen soll oder nicht.

Eindeutigkeit der Namen

Bei der Generierung der CON-, PTERM- und MUX-Anweisung ist zu beachten, dass das Namenstripel (*appliname* bzw. *ptermname*, *processorname*, *local_appliname*) innerhalb eines Generierungslaufs eindeutig sein muss.

B B B B B B B

Um die Generierung einer UTM-Anwendung in BS2000/OSD unabhängiger von der PDN-Generierung zu machen, können Sie Terminals in die Konfiguration aufnehmen, ohne ihren Typ explizit anzugeben. Dazu geben Sie im Operanden PTYPE den Wert *ANY an. openUTM übernimmt dann den Partnertyp (PTYPE) beim Verbindungsaufbau aus dem Benutzerdienstprotokoll (Connection Letter) und überprüft, ob der Typ unterstützt werden kann. Unterstützt openUTM den Typ nicht, dann wird der Verbindungswunsch abgelehnt.

```
PTERM␣      ptermname
            [ ,BCAMAPPL=local_appliname ]
            [ ,CONNECT={ YES | NO } ]
            [ ,ENCRYPTION-LEVEL={ NONE | 1 | 2 | 3 | 4 | TRUSTED } ]
            [ ,IDLETIME=time ]
            [ ,LISTENER-PORT=number ]
            [ ,LTERM=ltermname ]
            [ ,MAP={ USER | SYSTEM | SYS1[1] | SYS2[1] | SYS3[1] | SYS4[1]} ]
              ,PRONAM={ processorname | C'processorname'| *RSO[1] }
                        nur in BS2000/OSD Pflichtoperand
            [ ,STATUS={ ON | OFF } ]
            [ ,TERMN=termn_id ]
            [ ,USP-HDR={ NO | MSG | ALL } ]
```

B/X ***BS2000/OSD- und Unix-System-spezifischer Operand***

B/X `[ ,CID=printer_id ]`

B ***BS2000/OSD-spezifische Operanden***

B `[ ,PROTOCOL={ N | STATION } ]`

B `,PTYPE={ partnertyp | *ANY | *RSO }`

B `[ ,USAGE={ D | O } ]`

X/W ***Unix- und Windows-System-spezifische Operanden***

X/W `[ ,PTYPE={ partnertyp |`

X/W `PRINTER[2] | ( PRINTER ,printertype [ ,class ] )[2] } ]`

X/W `[ ,T-PROT=RFC1006 | SOCKET ]`

X/W `[ ,TSEL-FORMAT={ T | E | A } ]`

B [1] Diese Operandenwerte sind nur in BS2000/OSD erlaubt.

X [2] Diese Operandenwerte sind nur auf Unix-Systemen erlaubt.

ptermname Maximal 8 Zeichen langer Name des Clients oder Druckers

Der angegebene Name muss eindeutig sein und darf keinem weiteren Objekt der Namensklasse 3 zugeordnet sein. Siehe dazu auch Abschnitt „Eindeutigkeit der Namen und Adressen“ auf Seite 262.

Folgende Fälle sind zu unterscheiden:

Socket-Anwendungen (PTYPE=SOCKET)

– Wird die Verbindung von der lokalen Anwendung zum Client aufgebaut, dann kann *ptermname* beliebig gewählt werden. Er ist dann nur UTM-intern von Bedeutung, z.B. für die Administration.

– Soll die Verbindung von außen (Initiative beim Client) aufgebaut werden, dann muss *ptermname* die Portnummer im fernen System enthalten, über die der Client die Verbindung aufbaut. Als *ptermname* müssen Sie dann das Präfix „PRT“ gefolgt von fünf Ziffern (eventuell mit führenden Nullen) angeben, die die Portnummer bezeichnen. Z.B. müssen Sie *ptermname*=PRT08050 angeben, wenn die Verbindung über den Port 8050 aufgebaut werden soll.

  Der Verbindungsaufbau von außen zu einem bestimmten PTERM ist jedoch nur von Partnern möglich, die ihre Portnummer bei einem Verbindungsaufbau selbst setzen. openUTM tut dies nicht, d.h. für eine ferne UTM-Anwendung, die SOCKET-Verbindungen zur lokalen Anwendung aufbauen soll, können Sie keine PTERM-Anweisung absetzen. Sie muss sich über einen LTERM-Pool anschließen.

B B

Clients (≠ Socket-Anwendungen) und Drucker an UTM-Anwendungen unter BS2000/OSD

B B B

Es muss der Name übernommen werden, der bei der Generierung des Netzes für den Client/Drucker definiert wurde. Fragen Sie den Netzadministrator.

B B

Wenn ein RSO-Drucker beschrieben wird (PTYPE=*RSO), muss hier der Name des Druckers angegeben werden, wie er bei RSO definiert wurde.

X/W X/W

UPIC-Clients und TS-Anwendungen (≠ Socket-Anwendungen) an einer UTM-Anwendung unter Unix-, Windows-Systemen

X/W X/W X/W

Sie müssen für *ptermname* den T-Selektor angeben, der dem Client im fernen System zugeordnet ist, wenn Sie OPTION CHECK-RFC1006=YES setzen (siehe OPTION-Anweisung Seite 419).

X ***Drucker an einer UTM-Anwendung auf Unix-Systemen***

X Für Drucker wird als *ptermname* der Name der Spool Queue bzw. Druckergruppe angegeben, wie bei der Generierung des Unix-Systems festgelegt. Für die Ausgabe der Daten ruft der Druckerprozess (*utmprint*) das Skript *utmlp* auf (siehe PTYPE=PRINTER auf Seite 456).

X ***Terminals an einer UTM-Anwendung auf Unix-Systemen***

X Bei lokalen Terminals und Pseudoterminals muss in jeder PTERM-Anweisung für *ptermname* das Ergebnis des Kommandos `basename `tty`` angegeben werden, damit die UTM-Generierung mit der Generierung der Terminals auf dem Unix-System übereinstimmt.

X Es kann in Unix-Systemen vorkommen, dass die Standardzuordnung von *ptermname* durch openUTM nicht eindeutig ist. Abhängig von der Art der Netze, an die das System angeschlossen ist, können zwei oder mehr Pseudoterminals existieren, die sich in dem letzten Term des *tty* (hinter dem letzten Schrägstrich) nicht unterscheiden. Nur ein Terminal kann dann mit diesem *ptermname* die Verbindung zur Anwendung aufbauen. Die Verbindungsanforderung des zweiten Terminals wird von openUTM zurückgewiesen.

Beispiel

X Es können gleichzeitig die ttys `/dev/pts/12` und `/dev/inet/12` existieren. Fordert das Terminal `/dev/pts/12` die Verbindung zur Anwendung mit *ptermname* 12 zu einem Zeitpunkt an, zu dem das Terminal `/dev/inet/12` bereits mit der Anwendung verbunden ist, dann wird die Verbindungsanforderung von `/dev/pts/12` abgelehnt. Als *ptermname* werden die letzten beiden Teile der Ausgabe des *tty*-Kommandos angegeben; z.B. statt PTERM 12 die Anweisungen PTERM `pts/12` und PTERM `inet/12`.

X Stattdessen können Sie auch einen LTERM-Pool mit PTYPE=TTY generieren.

W ***Terminals an einer UTM-Anwendung auf Windows-Systemen***

W Es kann ein beliebiger *ptermname* angegeben werden.

BCAMAPPL=local_appliname

Name der lokalen UTM-Anwendung, wie er mit MAX ...,APPLINAME= oder mit der BCAMAPPL-Anweisung festgelegt wurde, siehe auch Seite 289. Beim Verbindungsaufbau zwischen Client/Drucker und UTM-Anwendung muss *local_appliname* als Partnername angegeben werden.

Bei Terminals und Druckern darf hier nur der in MAX ...,APPLINAME= definierte Name verwendet werden.

Für PTERMs mit PTYPE=SOCKET müssen Sie in *local_appliname* einen Namen angeben, der mit BCAMAPPL ... T-PROT=SOCKET generiert ist.

Der in der CLUSTER-Anweisung angegebene BCAMAPPL-Name ist hier nicht erlaubt.

Standardwert:
Wert aus MAX APPLINAME= (primärer Name der UTM-Anwendung)

Der Standardwert wird angenommen, wenn BCAMAPPL= nicht angegeben wird oder für BCAMAPPL= Leerzeichen angegeben werden.

B/X B/X B/X B/X CID=printer_id

gilt nur für Drucker, die einem Druckersteuer-LTERM zugeordnet werden. Über *printer_id* werden die Drucker am Druckersteuer-LTERM identifiziert. *printer_id* kann maximal 8 Zeichen lang sein.

B/X B/X Über das Druckersteuer-LTERM werden die Drucker, Drucker Queues und Druckaufträge administriert.

B/X B/X B/X B/X B/X Wird dem Drucker ein LTERM-Partner zugeordnet, für den mit LTERM ...,CTERM=*ltermname2* ein Druckersteuer-LTERM definiert wurde, dann ist dieser Drucker ebenfalls dem Druckersteuer-LTERM zugeordnet. Dabei muss die Kombination aus *ltermname2* des Druckersteuer-LTERMs und CID eindeutig sein.

B/X Standard: Dem Drucker wird keine CID zugeordnet.

CONNECT= gibt an, ob openUTM beim Anwendungsstart automatisch eine Verbindung zum Client bzw. Drucker aufbaut.

B B – Unter BS2000/OSD ist CONNECT= nur relevant für TS-Anwendungen, Terminals und Drucker.

X X – Unter Unix-Systemen ist CONNECT= nur relevant für TS-Anwendungen und Drucker.

W W – Unter Windows-Systemen ist CONNECT= nur relevant für TS-Anwendungen.

YES openUTM versucht, beim Start der Anwendung automatisch eine Verbindung aufzubauen.

Bei Druckern, die mit LTERM ...,PLEV > 0 generiert sind, versucht openUTM, die logische Verbindung erst aufzubauen, wenn der PLEV-Wert überschritten ist.

Kann die Verbindung zu einem Drucker oder einer TS-Anwendung nicht aufgebaut werden, versucht openUTM, die Verbindung in dem Zeitabstand aufzubauen, der in MAX ...,CONRTIME angegeben ist.

B B B *BS2000/OSD:*
Lässt sich die Verbindung zu einem Terminal nicht aufbauen, kann der Benutzer zu einem späteren Zeitpunkt die logische Verbindung aufbauen.

X X X X

Unix-Systeme:
Für Drucker erzeugt openUTM beim Start der Anwendung automatisch einen *ptermname* zugeordneten Druckerprozess, der Druckaufträge bearbeiten kann.

NO Beim Start der Anwendung versucht openUTM nicht eine Verbindung aufzubauen.

Für Clients mit PTYPE=UPIC-R und UPIC-L ist nur die Angabe CONNECT=NO erlaubt.

Standard: NO

X X X X X X

Unix-Systeme:
Für Drucker wird bei CONNECT=NO kein Druckerprozess erzeugt. Der Druckerprozess wird erst durch das Administrationskommando KDCPTERM ACT=C (oder KDCLTERM für den zugeordneten LTERM-Partner) eingerichtet. Dieser kann von den Workprozessen Druckaufträge entgegennehmen, die für die entsprechende Spool Queue bestimmt sind.

ENCRYPTION-LEVEL=
Nur relevant für UPIC-Clients (≥ V5.0), die die Verschlüsselung unterstützen, und unter BS2000/OSD zusätzlich für einige Terminalemulationen, die die Verschlüsselung unterstützen.

In ENCRYPTION-LEVEL legen Sie die minimale Verschlüsselungsebene für die Kommunikation mit dem Client fest.
Sie geben an, ob die UTM-Anwendung auf der Verbindung zum Client die Verschlüsselung der Nachrichten fordern soll oder nicht. Sie können den Client aber auch als „trusted“ Client definieren. Zur Verschlüsselung siehe auch Abschnitt „Nachrichten-Verschlüsselung auf Verbindungen zu Clients“ auf Seite 225.

Für die Verschlüsselung der Daten auf Verbindungen zum Client muss der Client unter openUTM-Client mit Trägersystem UPIC einer Version ≥ V5.0 mit Verschlüsselungsfunktionen ablaufen.

B B

Voraussetzung für den Einsatz der Verschlüsselung auf Verbindungen zwischen openUTM und Terminal-Emulationen ist VTSU-B ≥ V12.0C.

Folgende Angaben sind möglich:

NONE Die Verschlüsselung der Nachrichten, die zwischen dem Client und der UTM-Anwendung ausgetauscht werden, wird von openUTM **nicht** standardmäßig angefordert.
Passworte werden verschlüsselt übertragen, sofern beide Partner Verschlüsselung unterstützen, und zwar mit dem größten verfügbaren Schlüssel, d.h. mit AES, falls ein RSA-Schlüssel mit Länge ≥ 512 bit verfügbar ist, sonst mit DES. Der AES-Schlüssel wird wiederum mit dem höchsten verfügbaren RSA-Schlüssel verschlüsselt.
Vorgänge, für deren Vorgangs-TACs Verschlüsselung generiert wurde (siehe ENCRYPTION-LEVEL in der TAC-Anweisung ab Seite 488), können von diesem Client nur gestartet werden, wenn der Client beim Conversation- bzw. Verbindungsaufbau die Verschlüsselung aushandelt.

1 | 2 | 3 | 4 Die Verschlüsselung der Nachrichten, die zwischen dem Client und der UTM-Anwendung ausgetauscht werden, wird von openUTM standardmäßig angefordert. Der Wert (1 ... 4) gibt die Verschlüsselungsebene an. Der Client kann sich nur anschließen, wenn er mindestens diese Verschlüsselungsebene unterstützt. Andernfalls lehnt openUTM den Verbindungsaufbau zum Client ab.

Die Werte 1 bis 4 haben folgende Bedeutung:

1 Passworte und Ein-/Ausgabe-Nachrichten werden mit dem DES-Algorithmus verschlüsselt. Zum Austausch des DES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 200 bit verwendet.

2 Passworte und Ein-/Ausgabe-Nachrichten werden mit dem AES-Algorithmus verschlüsselt. Zum Austausch des AES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 512 bit verwendet.

3 Passworte und Ein-/Ausgabe-Nachrichten werden mit dem AES-Algorithmus verschlüsselt. Zum Austausch des AES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 1024 bit verwendet.

4 Passworte und Ein-/Ausgabe-Nachrichten werden mit dem AES-Algorithmus verschlüsselt. Zum Austausch des AES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 2048 bit verwendet.

ENCRYPTION-LEVEL=1 ... 4 ist für UPIC-Partner nur sinnvoll, wenn an Ihrem System die Encryption-Funktionalität von openUTM installiert ist. Anderfalls kann sich der Client nicht anschließen.

B Für VTSU-Partner wird die Verschlüsselung von VTSU durchgeführt.

Für die einzelnen Client-Typen gilt bezüglich der Verschlüsselungsebene:

– Für remote UPIC-Clients (PTYPE=UPIC-R) sind Verschlüsselungsebenen 1 bis 4 sinnvoll.

B
B
B

– Für Clients mit PTYPE= T9763 oder *ANY unter BS2000/OSD ist nur die Verschlüsselungsebene 1 sinnvoll (ENCRYPTION-LEVEL=1). Die Werte 2, 3 und 4 werden von KDCDEF ohne Meldung in 1 geändert.

X/W
X/W
X/W

– Bei lokalen UPIC-Clients (PTYPE=UPIC-L) einer Anwendung unter Unix- bzw. Windows-Systemen wird 1, 2, 3, oder 4 ohne Meldung in TRUSTED geändert.

– Wird 1 ... 4 für einen Partner eines anderen Typs angegeben, dann wird der Wert von openUTM ohne Meldung in NONE geändert.

Falls die Anwendung mit OPTION GEN-RSA-KEYS=NO generiert ist, dann werden beim KDCDEF-Lauf keine RSA-Schlüssel erzeugt. Um die Verschlüsselungsfunktionen nutzen zu können, müssen Sie die benötigten Schlüssel per Administration erzeugen (KC_ENCRYPT oder WinAdmin) oder per KDCUPD aus einer alten KDCFILE übertragen.

TRUSTED

Der Client ist ein vertrauenswürdiger „trusted“ Client.
Nachrichten zwischen Client und Anwendung werden nicht verschlüsselt.
Ein „trusted“ Client kann auch Vorgänge starten, deren Vorgangs-TACs Verschlüsselung erfordern (generiert mit TAC ENCRYPTION-LEVEL=1 | 2).

TRUSTED sollte nur gewählt werden, wenn der Client nicht für jedermann zugänglich ist und die Kommunikation über eine sichere Verbindung läuft.

X/W
X/W

Für lokale UPIC-Clients (UPIC-L) ist nur der Wert TRUSTED erlaubt. Andere Angaben werden von openUTM ohne Meldung in TRUSTED geändert.

Standard: NONE

IDLETIME=time

Darf nur für Dialog-Partner angegeben werden.
In *time* geben Sie die Zeit in Sekunden an, die openUTM außerhalb einer Transaktion, d.h. nach dem Ende einer Transaktion oder nach dem Anmelden, maximal auf eine Eingabe vom Client wartet. Bei Zeitüberschreitung baut openUTM die Verbindung zum Client ab. Ist der Client ein Terminal, dann wird vor dem Verbindungsabbau die Meldung K021 ausgegeben.

Der für *time* angegebene Wert darf nicht kleiner sein als die Werte der Timer TERMWAIT=*time* (Wartezeit zwischen Dialog-Ausgabe und -antwort innerhalb einer Mehrschritt-Transaktion; siehe Seite 401) und PGWTTIME=*time* (Zeit, die ein Teilprogramm nach einem blockierenden Aufruf auf das Eintreffen einer Nachricht wartet; siehe Seite 392), die Sie in der MAX-Anweisung festlegen.

Diese Funktion dient dazu den Datenschutz zu verbessern:
Vergisst ein Benutzer bei einer Unterbrechung oder der Beendigung seiner Arbeit am Terminal, sich abzumelden, dann wird die Verbindung zum Terminal bzw. zum Client nach Ablauf der Wartezeit automatisch abgebaut. Damit wird die Wahrscheinlichkeit für einen unberechtigten Zugang verringert.

Standard: 0 (= Warten ohne Zeitbegrenzung)
Maximalwert: 32767
Minimalwert: 60

Geben Sie einen Wert an, der kleiner als der Minimalwert ist, dann ersetzt KDCDEF den Wert durch den Minimalwert.

LISTENER-PORT=number

Portnummer für den Aufbau von TCP/IP-Verbindungen.

Bei Socket-Anwendungen (T-PROT=SOCKET) ist LISTENER-PORT= Pflichtoperand.

B *BS2000/OSD:*

B B B Unter BS2000/OSD darf LISTENER-PORT= nur für Socket-Anwendungen angegeben werden (PTYPE=SOCKET).
In diesem Fall ist LISTENER-PORT Pflichtoperand.

B B B Für *number* ist die Portnummer anzugeben, an der die Socket-Anwendung auf Verbindungsaufbauwünsche wartet, d.h. die Portnummer innerhalb des Partner-Rechners, über die die Socket-Anwendung adressiert wird.

B Es sind alle Portnummern erlaubt.

X/W *Unix-, Windows-Systeme*:

X/W X/W LISTENER-PORT ist relevant für remote UPIC-Clients (PTYPE=UPIC-R) und TS-Anwendungen (PTYPE=SOCKET oder APPLI).

X/W Erlaubte Werte:

X/W T-PROT=RFC1006: Portnummern 102 und 1025 - 32767.

X/W T-PROT=SOCKET: Portnummern zwischen 1 und 65535.

X/W X/W X/W

Der LISTENER-PORT wird bei T-PROT=SOCKET verwendet, um festzulegen, mit welcher Portnummer der Partner adressiert wird. Weitere Adressierungsinformationen sind unnötig.

X/W X/W X/W X/W

Standard: 0 (keine Portnummer)
Bei OPTION CHECK-RFC1006=YES muss für PTERMs mit PTYPE=APPLI oder SOCKET bei LISTENER-PORT eine Portnummer angegeben werden.

LTERM=ltermname

Name des LTERM-Partners, der dem Client/Drucker *ptermname* zugeordnet wird und über den sich der Client/Drucker an die UTM-Anwendung anschließt. *ltermname* ist max. 8 Zeichen lang.

Bei Clients vom Partnertyp PTYPE=SOCKET, APPLI oder UPIC-R **muss** der Operand LTERM= versorgt werden!

Die Zuordnung zwischen Client/Drucker und LTERM-Partner kann per Administrationskommando KDCSWTCH im Betrieb geändert werden, z.B. beim Ausfall eines Druckers. Es ist aber nicht möglich einem Drucker beispielsweise einen Dialog-LTERM-Partner (LTERM USAGE=D) zuzuordnen.

B/X B/X B/X B/X B/X

Für die Funktion Druckerbündel schreiben Sie mehrere PTERM-Anweisungen mit demselben *ltermname*. Ein Druckerbündel besteht damit aus mehreren Druckern, die einem LTERM-Partner zugeordnet sind (siehe auch Abschnitt „Druckerbündel generieren“ auf Seite 175). openUTM verteilt die Druckaufträge dabei zyklisch an die Drucker des Bündels.

MAP=

In MAP legen Sie fest, ob openUTM die Benutzer-Nachrichten, die mit dem Kommunikationspartner ausgetauscht werden, umcodieren soll oder nicht (EBCDIC <-> ASCII).
Benutzer-Nachrichten werden an der KDCS-Schnittstelle bei den Aufrufen zur Nachrichtenbehandlung (MPUT/MGET/FPUT/DPUT) im Nachrichtenbereich übergeben.

MAP= regelt die Konvertierung beim Austausch unformatierter Nachrichtern. Bei formatierten Nachrichten findet grundsätzlich keine Nachrichtenbehandlung statt.

MAP≠USER ist nur erlaubt:

B

– unter BS2000/OSD bei Socket-Anwendungen (PTYPE=SOCKET).

X/W X/W

– unter Unix- und Windows-Systemen bei allen TS-Anwendungen (PTYPE=SOCKET oder APPLI).

USER openUTM konvertiert die Daten des Nachrichtenbereichs nicht, d.h. die Nachrichten werden unverändert an die Partner-Anwendung übertragen bzw. die vom Partner empfangenen Nachrichten werden unverändert an das Teilprogramm übergeben.
Dabei ist zu beachten, dass bei TS-Anwendungen (Partner mit PTYPE=SOCKET oder APPLI) der Transaktionscode in der Benutzer-Nachricht enthalten ist. Er muss so kodiert sein wie ihn die UTM-Anwendung erwartet, d.h. unter BS2000/OSD in EBCDIC und unter Unix- und Windows-Systemen in ASCII.

Standard: USER

B SYSTEM | SYS | SYS1 | SYS2 | SYS3 | SYS4 (in BS2000/OSD)

B Der Parameter darf nur angegeben werden, wenn die von der Socket-Anwendung empfangenen Nachrichten nicht in EBCDIC codiert sind, bzw. wenn die Socket-Anwendung in ASCII codierte Nachrichten von der UTM-Anwendung erwartet.

B Geben Sie einen der obigen Werte an, dann konvertiert openUTM die Daten der Nachrichten vor dem Senden von EBCDIC nach ASCII und nach dem Empfangen von ASCII nach EBCDIC. Dabei geht openUTM davon aus, dass die Nachrichten nur abdruckbare Zeichen enthalten.

B Mit SYSTEM, SYS, SYS1, SYS2, SYS3, SYS4 geben Sie die Umsetztabelle an, die openUTM bei der Umcodierung verwenden soll.
Die Umsetztabellen müssen in dem Modul KDCEA definiert sein (siehe ASCII - EBCDIC Code-Konvertierung“ im openUTM-Handbuch „Anwendungen programmieren mit KDCS“).

B Die Angaben SYSTEM, SYS und SYS1 sind dabei synonym. Geben Sie einen dieser Werte an, dann verwendet openUTM für die Konvertierung die Standard-Codetabelle, die von EBCDIC nach 7-Bit ASCII umsetzt. Die Standard-Codetabelle (Tabelle 1) ist in KDCEA bereits vordefiniert und kann ohne weitere Vorbereitungen genutzt werden.

B Soll openUTM eine andere Umsetzung von EBCDIC nach ASCII durchführen, dann müssen Sie in KDCEA entsprechende Umsetztabellen selbst definieren. Sie können in KDCEA maximal drei eigene Umsetztabellen definieren (Tabelle 2 bis Tabelle 4).

B Geben Sie SYS2 an, dann konvertiert openUTM die Benutzer-Nachrichten mit Hilfe der Tabelle 2. Bei SYS3 verwendet openUTM die Tabelle 3 und bei SYS4 die Tabelle 4.

B Der Parameter ist nur für PTYPE=SOCKET zulässig.

X/W SYSTEM (in Unix-, Windows-Systemen)
X/W openUTM konvertiert die Daten des Nachrichtenbereichs vor dem Absenden von ASCII nach EBCDIC bzw. nach dem Empfangen von EBCDIC nach ASCII. Die Nachricht darf dabei nur abdruckbare Zeichen enthalten und muss im Zeilenmodus (KCMF = Leerzeichen) erzeugt werden.

X/W Der Parameter ist nur für PTYPE=SOCKET oder APPLI zulässig.

PRONAM={ processorname | C'processorname' }
maximal 8 Zeichen langer symbolischer Name des Partner-Rechners. Enthält *processorname* Sonderzeichen, muss er als Zeichenkette mit C'...' angegeben werden.

B *BS2000/OSD:*

B Es ist der Name anzugeben, der bei der Generierung des Netzes für den
B Partner-Rechner festgelegt wurde. Fragen Sie den Netzadministrator. Die
B Zuordnung von *ptermname* zu *processorname* muss eindeutig sein.

B Wird eine TS-Anwendung beschrieben, mit der die UTM-Anwendung über
B die Socket-Schnittstelle kommuniziert, dann müssen Sie für *processorname*
B die symbolische Adresse des Partner-Rechners angeben. Die Zuordnung
B symbolische Adresse zu realer IP-Adresse muss im Name Service des lo-
B kalen Systems (in der RDF-Datei) eingetragen sein. Einen Aliasnamen des
B Rechners dürfen Sie nicht angeben.

B Wenn ein RSO-Drucker beschrieben wird (PTYPE=*RSO), muss als
B Rechnername *RSO angegeben werden.

B Pflichtoperand.

B Sie müssen PRONAM nicht angeben, wenn Sie zuvor mit der DEFAULT-An-
B weisung einen Standardwert dafür festgelegt haben.

X/W *Unix-, Windows-Systeme:*

X/W Unter Unix- und Windows-Systemen darf PRONAM nur für remote UPIC-
X/W Clients (PTYPE=UPIC-R) und TS-Anwendungen (PTYPE=SOCKET oder
X/W APPLI) angegeben werden.

X/W Für die Angabe von *processorname* haben Sie zwei Möglichkeiten:

X/W – Sie geben den realen Hostnamen an, unter dem die IP-Adresse des
X/W Partner-Rechners im Name Service des lokalen Systems (z.B. der
X/W hosts-Datei) eingetragen ist. Einen Aliasnamen des Rechners dürfen
X/W Sie nicht angeben.

X/W – Sie geben den UTM-Hostnamen des Partner-Rechners an. Voraussetzung hierfür ist, dass Sie die Umgebungsvariable UTM_NET_HOSTNAME gesetzt und den UTM-Hostnamen in der Umwandlungsdatei festgelegt haben (siehe Abschnitt „Mapped Hostnamen verwenden (Unix- und Windows-Systeme)“ auf Seite 118).

X/W Die Angabe von *processorname* ist Pflicht.

X/W Standardwert: 8 Leerzeichen.

B PROTOCOL= Benutzerdienstprotokoll, das auf den Verbindungen zwischen der UTM-Anwendung und dem Client/Drucker gefahren werden soll.

B N Zwischen der UTM-Anwendung und dem Client/Drucker wird ohne Benutzerdienstprotokoll gearbeitet.

B Für UPIC-Clients (PTYPE=UPIC-R), TS-Anwendungen (PTYPE=SOCKET oder APPLI) oder Drucker, die über RSO angesprochen werden (PTYPE=*RSO), muss PROTOCOL=N gesetzt werden.

B Clients mit PROTOCOL=N können sich nicht über eine Multiplexverbindung (MUX-Anweisung) an die UTM-Anwendung anschließen.

B Haben Sie PTYPE=*ANY angegeben, dann ignoriert openUTM die Angabe PROTOCOL=N. openUTM setzt automatisch PROTOCOL=STATION ein.

B Standard: bei PTYPE=SOCKET, APPLI, UPIC-R oder *RSO

B STATION Zwischen der UTM-Anwendung und dem Client/Drucker wird mit dem Benutzerdienstprotokoll (NEABT) gearbeitet.

B Für Clients, die mit PTYPE=*ANY generiert sind, ist nur die Angabe PROTOCOL=STATION erlaubt. openUTM benötigt in diesem Fall das Benutzerdienstprotokoll (NEABT) zur Bestimmung des Gerätetyps des Clients oder Druckers.

B Für UPIC-Clients (PTYPE=UPIC-R), TS-Anwendungen (PTYPE=APPLI oder SOCKET) oder Drucker, die über RSO angesprochen werden (PTYPE=*RSO), ist nur die Angabe PROTOCOL=N erlaubt. Die Angabe von PROTOCOL=STATION wird ignoriert.

B Standard: STATION bei PTYPE≠SOCKET, APPLI, UPIC-R oder *RSO

PTYPE= Typ des Kommunikationspartners

B B Pflichtoperand in BS2000/OSD

B **PTYPE unter BS2000/OSD:**

B B B Für PTYPE müssen Sie entweder den Partnertyp *partnertyp* des Clients/Druckers oder den Wert *ANY oder den Wert *RSO für RSO-Drucker angeben.

B B B B B B partnertyp gibt explizit den Typ des Kommunikationspartners, d.h. des Clients oder Druckers, an. Der in *partnertyp* angegebene Wert muss mit der Angabe bei der PDN-Generierung übereinstimmen. Der Parameter PTYPE muss entweder hier angegeben oder in der DEFAULT-Anweisung explizit mit einem Wert versorgt werden. Die unterstützten Partnertypen entnehmen Sie der folgenden Tabelle:

	Partner	**PTYPE**	**TERMN**
B	DSS 9748	T9748 [2)]	FE
B	DSS 9749	T9749	FE
B	DSS 9750	T9750 [2)]	FE
B	DSS 9751	T9751	FE
B	DSS 9752	T9752	FF
B	DSS 9753	T9753	FE
B	DSS 9754	T9754	FI
B	DSS 9755	T9755 [3)]	FG
B	DSS 9756	T9756 [3)]	FG
B	DSS 9763	T9763	FH
B	DSS 9770	T9770	FK
B	DSS 9770R	T9770R	FK
B	FHS-DOORS-Frontend	DSS-FE	FH
B	DSS 3270 (IBM)	T3270	FL
B	DSS X28 (TELETYPE)	THCTX28	C5
B	DSS X28 (VIDEO)	TVDTX28	C6
B	Datenstation PT80	TPT80	C4
B	Drucker 9001	T9001	C7
B	Drucker 9002	T9002	FA
B	Drucker 9003	T9003	F9

Partner	PTYPE	TERMN
Drucker 9004	T9004	FD
Drucker 9001-3	T9001-3	CA
Drucker 9001-893	T9001-893	CB
Drucker 9011-18	T9011-18	CC
Drucker 9011-19	T9011-19	CD
Drucker 9012	T9012	CE
Drucker 9013	T9013	C9
Drucker 9021	T9021	CH
Drucker 9022	T9022	CF
Drucker 3287	T3287	CG
Intelligente Datenstation	THOST	A3
PDN-Anwendung	APDN [1)]	A2
Transportsystem-Anwendung , die keine Socket-Anwendung ist, z.B.: DCAM-, PDN, CMX- oder UTM-Anwendung.	APPLI	A1
Socket-Anwendung	SOCKET	A7
UPIC-Client	UPIC-R	A5

1) Eine PDN-Anwendung kann auch mit PTYPE=APPLI generiert werden. openUTM unterscheidet zwischen Anwendungen und Stationen. APDN wird als Station, APPLI als Anwendung behandelt.

2) Die PTYPEs T9748 und T9750 bezeichnen den gleichen Terminaltyp.

3) Die PTYPEs T9755 und T9756 bezeichnen den gleichen Terminaltyp.

In welcher VTSU-Version die einzelnen Terminals unterstützt werden, ist dem DCAM-, FHS- bzw. TIAM-Handbuch zu entnehmen. Wird ein Terminal von VTSU nicht unterstützt, so weist openUTM einen Verbindungsaufbau von diesem Terminal zurück. openUTM erzeugt die Meldungen K064 und K107.

B *ANY Durch die Angabe von PTYPE=*ANY wird ein VTSU-Client generiert. Der Client/Drucker wird ohne genaue Angabe des Gerätetyps in die Konfiguration aufgenommen. openUTM nimmt den Gerätetyp des Clients/Druckers beim Verbindungsaufbau aus dem Benutzerdienstprotokoll. Erst dann wird entschieden, ob der Partnertyp unterstützt wird.

B Der Vorteil von PTYPE=*ANY ist, dass Sie Clients in die Konfiguration aufnehmen können, ohne zu wissen, wie sie im PDN generiert sind. Darüber hinaus wird die Pflege der Konfiguration erleichtert, denn auch nach einer Umgenerierung im PDN können diese Clients weiterhin eine Verbindung zu der Anwendung aufbauen, ohne dass Sie die KDCDEF-Generierung ändern müssen.

B Wenn für Clients, die mit PTYPE=*ANY definiert wurden, explizit keine Terminalmnemonics generiert werden (Operand TERMN), wird beim Verbindungsaufbau die Standard-Terminalmnemonic des Partnertyps genommen.

B *RSO Mit PTYPE=*RSO können Drucker über RSO unterstützt werden. Anstatt eine Transportverbindung aufzubauen, reserviert openUTM den Drucker bei RSO und übergibt die zu druckende Nachricht an RSO.

X/W **PTYPE unter Unix-, Windows-Systemen:**

X/W partnertyp gibt explizit den Typ des Kommunikationspartners an, d.h. des Clients oder Druckers. Für *partnertyp* können Sie folgende Werte angeben:

TTY	Der Client ist ein Terminal. Standard: TTY
APPLI	Der Client ist eine Transportsystem-Anwendung, die nicht die Socket-Schnittstelle benutzt (z.B. UTM-, CMX-, DCAM-Anwendung).
SOCKET	Der Client ist eine Socket-Anwendung.
UPIC-L	Der Client ist ein lokaler UPIC-Client.
UPIC-R	Der Client ist ein ferner (remote) UPIC-Client (normalerweise auf einem anderen Rechner).

X PRINTER (nur unter Unix-Systemen)
Drucker ohne Zusatzparameter.
Für die Ausgabe der Daten ruft der Druckerprozess (*utmprint*) das Skript *utmlp* auf. Beim Aufruf werden zusätzlich zu den auszudruckenden Daten Parameter an *utmlp* übergeben. *utmlp* übergibt dann standardmäßig die Daten an das Kommando lp (Hinweise zum Skript *utmlp* siehe Seite 456).

(PRINTER, printertype,[class]) (nur unter Unix-Systemen)

Drucker mit erweiterten Parametern.
Für die Ausgabe der Daten ruft der Druckerprozess (*utmprint*) das Skript *utmlp* auf. Beim Aufruf werden zusätzlich zu den auszudruckenden Daten die erweiterten Parameter an *utmlp* übergeben. *utmlp* übergibt die Daten mit dem Wert von *class* als *destination* an das Kommando lp (Hinweise zum Skript *utmlp* siehe unten).

printertype
bezeichnet den Druckertyp, für den Druckausgaben bestimmt sind.
Sind Sonderzeichen im Parameterwert von *printertype* enthalten, müssen Sie den Wert in einfache Hochkommata einschließen.
Maximale Länge von *printertype*: 8 Zeichen

class
Name der Druckergruppe (printer class).
Enthält der Name der Druckergruppe Sonderzeichen, müssen Sie den Wert in einfache Hochkommata einschließen.
Maximale Länge: 40 Zeichen
Standardwert: Wert von *ptermname*

Hinweise zum Skript utmlp

– Das Skript *utmlp* wird mit ausgeliefert. Sie finden es im Verzeichnis $UTMPATH/shsc.

– Die Parameter sind im Skript selbst dokumentiert.

– Der Zugriff auf das Skript erfolgt zur Laufzeit über $PATH.

– Sie können das Skript editieren, um z.B. die Nachricht vor dem Ausdrucken anzupassen oder über das Netzwerk zu drucken.

– Bei erfolgreicher Verarbeitung liefert das Skript den Exitcode 0 (Null). Bei einem Exitcode ungleich 0 wird die Verbindung zum Druckerprozess beendet und nach einem erneuten Verbindungsaufbau die Ausgabe wiederholt.

Bei Clients vom Typ APPLI, SOCKET oder UPIC-R kann es vorkommen, dass eine Verbindung zu einem Client aus Sicht von openUTM noch besteht, der Client aber keine Verbindung mehr zur Anwendung hat und deshalb versucht, sie neu aufzubauen. Dazu sendet er einen Connection Request an openUTM. openUTM baut daraufhin die „bestehende“ Verbindung ab.

Für Clients vom Typ APPLI oder SOCKET initiiert openUTM anschließend automatisch den Aufbau einer neuen Verbindung.

Für UPIC-Clients muss die Initiative zum Aufbau einer neuen Verbindung vom UPIC-Client ausgehen.

STATUS= gibt an, ob der Client/Drucker beim Anwendungsstart gesperrt ist.

ON Der Client oder Drucker ist nicht gesperrt. Sofern der LTERM-Partner, über den sich der Client/Drucker an die UTM-Anwendung anschließt, nicht gesperrt ist, können Verbindungen aufgebaut werden oder es bestehen bereits Verbindungen.

Standard: ON

OFF Der Client oder Drucker ist gesperrt. Es können keine Verbindungen zwischen Client/Drucker und lokaler Anwendung aufgebaut werden. Der Administrator kann den Client/Drucker freigeben.

TERMN=termn_id

Kennzeichen für die Art des Clients, das openUTM dem Anwendungsprogramm im Feld KCTERMN des KB-Kopfes zur Verfügung stellt. *termn_id* ist max. 2 Zeichen lang. *termn_id* wird nicht von openUTM abgefragt, es kann vom Benutzer zur Auswertung verwendet werden.

Standardwerte:

Wird der Operand nicht angegeben, so trägt openUTM in KCTERMN das Standard-Kennzeichen für den Partnertyp ein, der beim Operanden PTYPE angegeben wird. Der Benutzer kann jedoch auch andere Werte wählen.

B – *BS2000/OSD:*
B Die Standard-Kennzeichen sind der Partnertyp-Tabelle beim Operanden PTYPE= auf Seite 454 zu entnehmen.
B Wird TERMN bei Clients, die mit PTYPE=*ANY generiert sind, nicht explizit angegeben, dann trägt openUTM das Kennzeichen für die Terminalmnemonic erst beim Aufbau der Verbindung in KCTERMN ein. Es
B wird die Standard-Terminalmnemonic des Typs eingetragen, der im
B Benutzerdienstprotokoll der Verbindungsanforderung enthalten ist.

X/W X/W – *Unix-, Windows-Systeme*
Die Standard-Kennzeichen sind der folgenden Tabelle zu entnehmen.

X/W	**PTYPE**	**TERMN**
X/W	TTY	F1
X	PRINTER	F2
X	PRINTER,fotyp,class	F2
X/W	APPLI	A1
X/W	UPIC-L	A2
X/W	UPIC-R	A5
X/W	SOCKET	A7

X/W T-PROT= Adressformat, mit dem sich der Client beim Transportsystem anmeldet.
X/W Ist nur relevant bei PTYPE=SOCKET, APPLI und UPIC-R.

X/W Informationen zu den folgenden Adressformaten finden Sie im CMX-Benut-
X/W zerhandbuch oder in der Online-Hilfe zu PCMX.

X/W RFC1006 Adressformat RFC1006

X/W SOCKET Die Kommunikation erfolgt über die Socket-Schnittstelle.
X/W SOCKET darf nur angegeben werden, wenn der Name der lokalen UTM-
X/W Anwendung, den Sie in BCAMAPPL angeben, mit T-PROT=SOCKET ge-
X/W neriert ist.
X/W Die Angabe einer Portnummer im Operanden LISTENER-PORT ist Pflicht.

X/W Der Standardwert für T-PROT ist abhängig von der Angabe bei PTYPE:

X/W T-PROT=RFC1006, falls PTYPE=APPLI oder UPIC-R
X/W T-PROT=SOCKET, falls PTYPE=SOCKET

X/W TSEL-FORMAT=
X/W Formatindikator des T-Selektors in der Transportadresse des Client.
X/W Ist nur relevant bei PTYPE=SOCKET, APPLI oder UPIC-R.

X/W Der Formatindikator gibt die Codierung des T-Selektors im Transportproto-
X/W koll an. Nähere Informationen finden Sie im CMX-Benutzerhandbuch bzw.
X/W in der Online-Hilfe zu PCMX.

X/W T TRANSDATA-Format

X/W E EBCDIC-Zeichenformat

X/W A ASCII-Zeichenformat

X/W
X/W

Für den Betrieb über RFC1006 wird empfohlen, explizit einen Wert für TSEL-FORMAT anzugeben.

B
B

USAGE= gibt an, ob der Kommunikationspartner ein Dialog-Partner oder ein reines Ausgabemedium ist.

B
B
B

D Der Client ist ein Dialog-Partner. Auf den Verbindungen zwischen Client und lokaler Anwendung kann sowohl die Anwendung als auch der Client Nachrichten senden.

B

UPIC-Clients (PTYPE=UPIC-R) sind immer Dialog-Partner.

B
B

Ein LTERM-Partner mit USAGE=D darf nicht einem Client mit USAGE=O zugewiesen werden.

B

Standard: bei PTYPE=SOCKET, APPLI, UPIC-R oder einem Terminal.

B
B

O Der Kommunikationspartner ist ein Drucker. Es können nur Nachrichten von der Anwendung zum Drucker gesendet werden.

B
B
B

Standard:
Für Partner, die mit PTYPE=*RSO generiert sind, darf nur USAGE=O generiert werden.

USP-HDR= Mit diesem Parameter wird gesteuert, für welche Ausgabe-Nachrichten openUTM auf dieser Verbindung einen UTM-Socket-Protokoll-Header aufbauen soll.
Eine Beschreibung des USP-Headers finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“.

Dieser Parameter ist nur von Bedeutung für PTERMs mit PTYPE=SOCKET.

NO openUTM erzeugt für keine Ausgabe-Nachricht einen UTM-Socket-Protokoll-Header.

Standard.

MSG Nur bei der Ausgabe von K-Meldungen erzeugt openUTM einen UTM-Socket-Protokoll-Header und stellt diesen der Nachricht voran.

ALL Bei allen Ausgabe-Nachrichten (Dialog, Asynchron, K-Meldungen) erzeugt openUTM einen UTM-Socket-Protokoll-Header und stellt diesen der Nachricht voran.

QUEUE - Tabelleneinträge für Temporäre Queues reservieren

Mit der Steueranweisung QUEUE legen Sie die maximale Anzahl Temporärer Queues fest, die in der Anwendung gleichzeitig existieren können. In der KDCFILE werden entsprechend viele Tabelleneinträge für Temporäre Queues reserviert. Darüber hinaus können Sie Standardeinstellungen für diese Queues definieren.

Temporäre Queues eignen sich z.B. für die Kommunikation zwischen zwei Vorgängen. Sie können im Betrieb dynamisch mit den KDCS-Aufrufen QCRE und QREL erzeugt und gelöscht werden.

Die QUEUE-Anweisung dürfen Sie innerhalb eines Generierungslaufs höchstens einmal angeben.

Näheres zu Queues und den damit möglichen Anwendungen finden sie im openUTM-Handbuch „Konzepte und Funktionen“.

```
QUEUE␣        NUMBER=queue-number
              [ ,QLEV=queue_level_number ]
              [ ,QMODE = { STD | WRAP-AROUND } ]
```

NUMBER=queue-number
: Legt die maximale Anzahl Temporärer Queues fest, die während eines Anwendungslauf zu einer Zeit existieren können.

  Minimalwert: 1
  Maximalwert: 500.000

QLEV=queue_level_number
: (**Q**ueue **Lev**el)
  Legt den Standardwert für die maximale Anzahl von Nachrichten fest, die gleichzeitig in einer Temporären Queue stehen dürfen.
  Die maximale Nachrichten-Anzahl kann Queue-spezifisch beim Erzeugen der Queue mit dem KDCS-Aufruf QCRE festgelegt werden (Parameter KCLA). Der mit QLEV= generierte Standardwert wird angewendet, wenn im Parameter KCLA der Wert 0 angegeben wird.

  QLEV=32767 bedeutet, dass die Anzahl der Nachrichten in der Queue standardmäßig nicht beschränkt wird.

  Standardwert: 32767 (d.h. unbeschränkte Queue-Länge)
  Minimalwert: 1
  Maximalwert: 32767 (d.h. unbeschränkte Queue-Länge)

QMODE = (**Q**ueue **Mode**)
Bestimmt das Verhalten von openUTM für den Fall, dass bereits die maximal erlaubte Anzahl von Nachrichten in einer Temporären Queue gespeichert und somit der Queue-Level erreicht ist.
Der hier generierte Wert wird beim dynamischen Erzeugen einer Temporären Queue angewendet, sofern im KDCS-Aufruf QCRE (Parameter KCQMODE) kein anderer Wert angegeben wird.

STD Wenn der Queue-Level einer Temporären Queue erreicht ist, dann lehnt openUTM weitere Nachrichten für diese Queue mit einem negativen Returncode ab.

Standard: STD

WRAP-AROUND
openUTM nimmt auch dann noch Nachrichten für die Temporäre Queue an, wenn der Queue-Level bereits erreicht ist. Beim Schreiben einer Nachricht in die Queue löscht openUTM dann die älteste der bereits in der Queue stehenden Nachrichten.

REMARK - Kommentarzeile einfügen

Mit der Steueranweisung können Sie einen Kommentar in die KDCDEF-Steueranweisungen einfügen. Kommentarzeilen sind immer einzeilig.

```
REMARK␣      comment
```

comment Als Kommentar können Sie eine beliebige Zeichenfolge angeben.

Eine Kommentarzeile kann auch durch einen Stern * in der Spalte 1 erzeugt werden.

RESERVE - Tabellenplätze für UTM-Objekte reservieren

Wenn Sie die Funktionen der Programmschnittstelle KDCADMI zur dynamischen Konfiguration im Anwendungsbetrieb nutzen oder dynamisch Objekte mittels des Administrationsarbeitsplatzes WinAdmin hinzufügen wollen, müssen Sie bei der KDCDEF-Generierung mit der RESERVE-Anweisung Tabellenplätze in den Objekttabellen von openUTM reservieren.

Was Sie außerdem bei der dynamischen Konfiguration beachten müssen, ist im Kapitel „Konfiguration einer Anwendung dynamisch ändern“ auf Seite 577 beschrieben.

Die RESERVE-Anweisung darf pro Objekttyp nur einmal angegeben werden. Für eine RESERVE-Anweisung mit OBJECT=ALL gilt:

- Nach RESERVE OBJECT=ALL dürfen keine weiteren RESERVE-Anweisungen eingegeben werden.
- Vor RESERVE OBJECT=ALL sind Objekt-spezifische RESERVE-Anweisungen erlaubt. Diese Objekt-spezifischen RESERVE-Anweisungen haben dann Vorrang.

Das Generierungstool KDCDEF reserviert auf Grund interner Abhängigkeiten evtl. mehr Objekte als in RESERVE-Anweisungen angegeben. Die genaue Zahl der Objekte der jeweils reservierten Einträge für einen Objekttyp wird in einer Meldung ausgegeben.

```
RESERVE␣      OBJECT={ ALL [ ,CARDS=percent1[1] ] [ ,PRINCIPALS=percent2[1] ] |
                       CON |
                       KSET |
                       LSES |
                       LTAC |
                       LTERM |
                       PROGRAM |
                       PTERM |
                       TAC |
                       USER [ ,CARDS=percent1[1] ] [ ,PRINCIPALS=percent2[1] ] }

                  [ { ,NUMBER=number | ,PERCENT=percent3 } ]
```

B [1] nur in BS2000/OSD erlaubt

OBJECT= Es werden Tabellenplätze für Objekte des angegebenen Typs reserviert, die bei Bedarf dynamisch in die Konfiguration eingetragen werden können.

ALL [,CARDS=percent1] [,PRINCIPALS=percent2]

B (CARDS= und PRINCIPALS= nur in BS2000/OSD erlaubt)
Es können Tabellenplätze für Objekte der Objekttypen CON, KSET, LSES, LTAC, LTERM, PROGRAM, PTERM, TAC und USER reserviert werden, die dann dynamisch eingetragen werden.

Für Objekte vom Objekttyp USER wird mit CARDS=*percent1* festgelegt, dass maximal *percent1* Prozent der dynamisch eingetragenen USER mit Ausweiskarte definiert werden können. Mit PRINCIPALS=*percent2* wird festgelegt, dass maximal *percent2* Prozent der dynamisch eingetragenen USER mit Kerberos-Authentisierung definiert werden können.

Standard für *percent1*/*percent2*: 0%, d.h. es können keine Benutzer mit Ausweiskarte bzw. Kerberos-Authentisierung dynamisch eingetragen werden. Maximalwert für *percent1*/*percent2*: 100%

CON Es werden Tabelleneinträge für die Transportverbindungen zu LU6.1-Partner-Anwendungen reserviert, d.h. für Objekte vom Typ CON.

KSET Es werden Tabelleneinträge für Keysets reserviert, d.h. für Objekte vom Typ KSET.

LSES Es werden Tabelleneinträge für LU6.1-Sessionnamen reserviert, d.h. für Objekte vom Typ LSES.

LTAC Es werden Tabelleneinträge für lokale Servicenamen reserviert, über die Service-Programme in Partner-Anwendungen gestartet werden können. Das sind Objekte vom Typ LTAC.

LTERM Es werden Tabellenplätze für Objekte vom Typ LTERM reserviert.

Bitte beachten Sie, dass die folgenden Objektkomponenten statisch generiert werden müssen und nicht dynamisch eingetragen werden können:

- Einem LTERM-Partner ohne Autosign-USER mit dem Namen eines statisch generierten USERs kann dynamisch kein Client mit PTYPE=APPLI zugeordnet werden.
- Einem LTERM-Partner mit statisch generiertem Autosign-USER kann dynamisch kein Client mit PTYPE=APPLI zugeordnet werden.
- Formatierungssystem bei der Verwendung von Formaten
- Anmelde-Vorgang bei #-Formaten

PROGRAM

Es werden Tabellenplätze für Objekte vom Typ PROGRAM reserviert. Objekte vom Typ PROGRAM können nur dynamisch in eine Anwendung eingetragen werden, die mit Lademodulen (BS2000/OSD), Shared Objects (Unix-Systeme) bzw. DLLs (Windows-Systeme) generiert ist.

Bitte beachten Sie, dass die folgenden Objektkomponenten statisch generiert werden müssen und nicht dynamisch eingetragen werden können:

- Programmiersprachen (PROGRAM ...,COMP=) müssen durch eine PROGRAM-Anweisung statisch generiert werden.
- Bei ILCS-fähigen Sprachen (COMP=ILCS) genügt die statische Generierung eines ILCS-Programms.

B
B
– LOAD-MODULEs bei PROGRAM müssen durch eine LOAD-MODULE-Anweisung statisch generiert werden.

B
B
B
– Für Anwendungen, die im BS2000/OSD ohne Lademodule generiert sind, müssen die PROGRAM-Namen, die beim Neueintragen eines TACs eingetragen werden, statisch generiert werden.

X/W
X/W
X/W
– Für Anwendungen, die ohne Shared Objects/DLLs generiert sind, müssen die Programmnamen, die beim Eintragen eines TACs angegeben werden, statisch generiert sein.

PTERM Es werden Tabellenplätze für Objekte vom Typ PTERM reserviert.

Für jeden Client mit PTYPE=APPLI, SOCKET, UPIC-R oder UPIC-L wird dabei von openUTM implizit ein USER erzeugt. Wenn solche Clients dynamisch generiert werden sollen, muss dies für OBJECT=USER bei den Angaben für NUMBER= bzw. PERCENT= berücksichtigt werden.

Bitte beachten Sie, dass BCAMAPPL-Namen statisch generiert werden müssen und nicht dynamisch eingetragen werden können.

TAC Es werden Tabellenplätze für Objekte vom Typ TAC reserviert.

Bitte beachten Sie, dass die folgenden Objektkomponenten statisch generiert werden müssen und nicht dynamisch eingetragen werden können:

– TAC-Klassen
– mindestens ein X/Open-TAC muss statisch generiert werden, wenn TACs für X/Open-Teilprogramme dynamisch erzeugt werden sollen.

USER [,CARDS=percent1] [,PRINCIPALS=percent2]

B
(CARDS= und PRINCIPALS= nur in BS2000/OSD erlaubt)
Es werden Tabellenplätze für Objekte vom Typ USER reserviert.

Wenn für eine Anwendung keine Benutzerkennungen generiert wurden, d.h. die Generierung enthält keine USER-Anweisungen, können keine Tabellenplätze für Objekte vom Typ USER reserviert werden, da KDCDEF in diesem Fall für jedes reservierte Objekt vom Typ LTERM intern bereits ein Objekt vom Typ USER reserviert. Die Anzahl der von KDCDEF auf diese Weise reservierten USER wird in einer Meldung ausgegeben.

B
B
B
B
B
Mit CARDS=*percent1* legen Sie fest, dass maximal *percent1* Prozent der dynamisch eingetragenen USER mit Ausweiskarte definiert werden können. Mit PRINCIPALS=*percent2* wird festgelegt, dass maximal *percent2* Prozent der dynamisch eingetragenen USER mit Kerberos-Authentisierung definiert werden können.

B B B

Standard für *percent1*/*percent2*: 0%, d.h. es können keine Benutzer mit Ausweiskarte bzw. Kerberos-Authentisierung dynamisch eingetragen werden. Maximalwert für *percent1*/*percent2*: 100%

B B

Bitte beachten Sie, dass die folgenden Objektkomponenten statisch generiert werden müssen und nicht dynamisch eingetragen werden können:

B B

– Formatierungssystem bei der Verwendung von Formaten
– Anmelde-Vorgang bei #-Formaten

Für alle TS-Anwendungen (PTYPE=APPLI/SOCKET) und UPIC-Clients (PTYPE=UPIC-R) wird von UTM intern eine Benutzerkennung erzeugt. Daher muss die NUMBER- bzw. PERCENT-Angabe entsprechend erhöht werden, falls diese PTERMs dynamisch eingetragen werden sollen.

NUMBER=number

Es können maximal *number* viele Objekte des angegebenen Objekttyps dynamisch eingetragen werden.

Bei OBJECTS=ALL können jeweils bis zu *number* viele Objekte der im Syntaxdiagramm angeführten Objekttypen dynamisch eingetragen werden.

– NUMBER=0
  bedeutet, dass die Anzahl der Objekte des angegebenen Objekttyps dynamisch bis auf den maximal generierbaren Wert erhöht werden kann. Dieser Wert kann der Übersicht über die Maximalwerte für die Anzahl der generierbaren Namen im Abschnitt „Anzahl der Namen“ auf Seite 259 entnommen werden.

– NUMBER≠0
  bewirkt, dass der Speicherverbrauch der UTM-Anwendung reduziert wird. Verletzt die Anzahl der zu reservierenden Objekte dabei einen Maximalwert für die Anzahl der generierbaren Namen (siehe Abschnitt „Anzahl der Namen“ auf Seite 259), so wirkt die Anweisung wie NUMBER=0.

Minimalwert: 0
Maximalwerte:

32.000	bei LTAC, KSET, TAC und PROGRAM
65.000	bei CON, LSES
500.000	bei USER, LTERM, PTERM

Außerdem gilt:

– Die Summe der reservierten Einträge für einen Objekttyp und die Anzahl der statisch generierten Namen der zugehörigen Namensklasse darf die maximale Anzahl der für diese Namensklasse zulässigen Einträge nicht überschreiten (siehe Abschnitt „Anzahl der Namen“ auf Seite 259).
– Die Summe von reservierten CONs und PTERMs darf nicht größer sein als 500.000.
– Die Summe von reservierten LSES und USER darf nicht größer sein als 500.000.

Am Ende des KDCDEF-Laufs wird für jeden Objekttyp die Anzahl der für diesen Typ resevierten Einträge mit der Meldung K502 ausgegeben.

PERCENT=percent3

Gibt die Anzahl der dynamisch eintragbaren Objekte des angegebenen Typs in *percent3* als Prozentwert der Gesamtanzahl der statisch generierten Objekte dieses Typs an.

Die Angabe in Prozent hat den Vorteil, dass bei jeder Generierung automatisch bei nicht geänderten RESERVE-Anweisungen die Anzahl der dynamisch eintragbaren Objekte im gleichen Verhältnis wächst, wie die Anzahl der statisch generierten Objekte des jeweiligen Typs.

Die Angabe PERCENT=*percent3* hat die gleiche Wirkung wie die Angabe NUMBER=*number* mit entsprechendem Wert von *number*, d.h. PERCENT=0 wirkt wie NUMBER=0.

Mit PERCENT≠0 kann der Speicherverbrauch der UTM-Anwendung reduziert werden. Verletzt die Anzahl der zu reservierenden Objekte einen Maximalwert für die Anzahl der generierbaren Namen, so wirkt die Anweisung wie PERCENT=0 (siehe Abschnitt „Anzahl der Namen“ auf Seite 259).

Standard: 10
Minimalwert: 0
Maximalwert: wird beschränkt durch die Anzahl generierbarer Namen

RMXA - Namen für XA-(Datenbank-)Anschluss definieren (Unix-, Windows-Systeme)

X/W Über die von X/Open genormte XA-Schnittstelle kann openUTM mit jedem Resource Manager gekoppelt werden, der diese Schnittstelle unterstützt, z.B. die Datenbanksysteme INFORMIX und Oracle. openUTM unterstützt die Kopplung über die XA-CAE-Schnittstelle (CAE-Specification).

X/W Für jeden Resource Manager muss eine RMXA-Anweisung geschrieben werden.

X/W Mit welcher Datenbank openUTM über den Resource Manager gekoppelt wird, ist i.A. abhängig von den Startparametern für den Resource Manager.

X/W
```
RMXA␣          XASWITCH=name
               [ ,SPEC=C ]
               [ ,DLLIMPORT={ YES | NO } ]
```

X/W XASWITCH=name

Name der *xa_switch_t*-Struktur des Resource Managers, die openUTM bekannt gemacht wird. *name* ist im jeweiligen Resource Manager fest vorgegeben. Siehe dazu auch Abschnitt „Datenbankkopplung definieren“ auf Seite 232.

X/W Pflichtoperand.

X/W SPEC=C Es wird die CAE-Spezifikation der XA-Schnittstelle benutzt.

X/W DLLIMPORT= gibt an, wie die Adressierung der *xa_switch_t*-Struktur des Resource Managers erfolgt.

W YES ist nur in openUTM auf Windows-Systemen erlaubt.
Die *xa_switch_t*-Struktur wird mit `dllimport` adressiert.
Für die Kopplung mit Oracle auf Windows-Systemen müssen Sie DLLIMPORT=YES generieren.

X/W NO Die *xa_switch_t*-Struktur wird mit `extern` adressiert.

X/W Standard: NO

ROOT - Namen für ROOT-Tabellen-Source vergeben

Wenn eine ROOT-Tabellen-Source erzeugt werden soll, muss eine ROOT-Anweisung angegeben werden. Die Anweisung kann entfallen, wenn nur die KDCFILE erstellt werden soll, d.h. in der OPTION-Anweisung ist weder GEN=ALL noch GEN=ROOTSRC angegeben. Die ROOT-Anweisung darf nur einmal angegeben werden.

```
ROOT␣        rootname
```

rootname Pflichtoperand, anzugeben ist:

B B *BS2000/OSD:*
CSECT-Name des einzubindenden KDCROOT-Tabellenmoduls.

B B B B Bei Verwendung der ROOT-Nachladetechnik wird dieses Modul beim Start der Anwendung aus der Bibliothek nachgeladen, die beim Startparameter TABLIB=*libname* angegeben wurde. Ist dies nicht der Fall, muss das Modul statisch eingebunden werden.

X/W X/W X/W X/W X/W *Unix-, Windows-Systeme:*
Namensteil der Datei, in der die ROOT-Tabellen-Source als C/C++ Programm abgelegt wird. *rootname* ist alphanumerisch anzugeben und kann max. 8 Zeichen lang sein. Der vollqualifizierte Name lautet: *filebase/rootname*`.c` (Unix-Systeme) bzw. *filebase**rootname*`.c` (Windows-Systeme).

SATSEL - SAT-Protokollierung steuern (BS2000/OSD)

B Mit der SATSEL-Anweisung wird festgelegt, welche Ereignisse aus welcher UTM-Ereignisklasse mit SAT protokolliert werden sollen (Preselection der zu protokollierenden Ereignisse). Zur Spezifikation der zu protokollierenden Ereignisse geben Sie die Ereignisklasse an, zu der die Ereignisse gehören. Innerhalb einer Ereignisklasse schränken Sie die Protokollierung ein, indem Sie angeben, ob nur erfolgreiche Bearbeitungsergebnisse oder nur nicht erfolgreiche Bearbeitungsergebnisse protokolliert werden sollen.

B Die SATSEL-Anweisung kann mehrfach angegeben werden. Wenn eine Ereignisklasse in mehreren SATSEL-Anweisungen erscheint, gelten für die Art der Protokollierung die Angaben in der ersten Anweisung.

B Soll die SAT-Protokollierung mit dem Start der Anwendung beginnen, so muss die SAT-Protokollierung per Generierung eingeschaltet werden (MAX ...,SAT=ON).

B Wenn MAX ...,SAT=OFF generiert wurde, können Sie die Auswahl der zu protokollierenden Ereignisse mit SATSEL auch bei ausgeschalteter SAT-Protokollierung in die Generierung einbringen. Die SATSEL-Anweisungen werden dann nicht wirksam. Auf diese Art können Sie jedoch eine Vorbelegung für die SAT-Protokollierung erzeugen. Die SAT-Protokollierung kann im laufenden Betrieb bei Bedarf eingeschaltet werden (Administrationskommando KDCMSAT).

B Die Art der Protokollierung der Ereignisse können Sie auch in den Anweisungen USER (Benutzer-spezifisch) und TAC (TAC-spezifisch) mit dem Operanden SATSEL festlegen. Werden in verschiedenen Anweisungen Angaben gemacht, so gilt Folgendes:

- B Eine Protokollierung erfolgt, sobald in einer Anweisung die Protokollierung eingeschaltet wurde. Dann ist auch die Art der Protokollierung (SUCC, FAIL oder BOTH) eindeutig.
- B Die SAT-Protokollierung kann in mehreren Anweisungen eingeschaltet werden (Anweisungen SATSEL, USER, TAC). Werden bei den verschiedenen Anweisungen verschiedene Arten der Protokollierung angegeben, bildet openUTM aus den Angaben die Obermenge für die Protokollierung. Die Obermenge ergibt sich aus der Oder-Verknüpfung der einzelnen Einstellungen. Ausnahme: Wird für eine Ereignisklasse der SATSEL-Anweisung OFF angegeben, so erfolgt dafür keine Protokollierung, auch wenn sie in der USER- oder TAC-Anweisung eingeschaltet wird. Einen Überblick über die möglichen Kombinationen von SAT-Protokollierungsbedingungen und deren Ergebnis finden Sie im openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“.
- B Jedes zu protokollierende Ereignis ist (mit Ausnahme von SIGN, CHANGE-PW) einem USER und einem TAC zuzuordnen. Daher kann die Protokollierung eines Ereignisses über die SATSEL-Anweisung (Protokoll für ein Ereignis einschalten) oder den SATSEL-Operanden der USER- bzw. TAC-Anweisung eingeschaltet werden.

B B B

Nähere Informationen zur SAT-Protokollierung sowie über mögliche Kombinationen von SAT-Protokollierungsbedingungen und deren Ergebnis finden Sie im openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“

B
```
SATSEL␣      { BOTH | SUCC | FAIL | NONE | OFF }
```
B
```
             , EVENT=(event1, event2, ...)
```

B B BOTH Für die bei EVENT angegebenen Ereignisklassen werden sowohl erfolgreiche als auch nicht erfolgreiche Ereignisse protokolliert.

B B B SUCC Für die bei EVENT angegebenen Ereignisklassen werden erfolgreiche Ereignisse protokolliert. Zusätzlich erfolgt eine SAT-Protokollierung entsprechend der Angaben für SATSEL in der USER- und TAC-Anweisung.

B B B FAIL Für die bei EVENT angegebenen Ereignisklassen werden nicht erfolgreiche Ereignisse protokolliert. Es erfolgt eine SAT-Protokollierung entsprechend den Angaben für SATSEL in der USER- und TAC-Anweisung.

B B B NONE Keine Ereignis-spezifische Auswahl für die SAT-Protokollierung. Eine SAT-Protokollierung erfolgt nur, wenn eine benutzer- und/oder TAC-spezifische SAT-Protokollierung eingeschaltet ist.

B B B B B B B OFF Für die in EVENT angegebenen Ereignisklassen werden keine Ereignisse protokolliert, auch wenn die SAT-Protokollierung im USER- oder TAC-Kommando eingeschaltet wurde. Dies ermöglicht einen generellen Verzicht auf die Protokollierung von Ereignissen, die als nicht sicherheitsrelevant angesehen werden (z.B. Zugriffe auf TLS-Bereiche) und damit eine sinnvolle Beschränkung der Protokolldatenmenge.

B B B EVENT=(event1, event2, ...)
legt die Ereignisklassen fest, die protokolliert werden sollen. Folgende Ereignisklassen können Sie auswählen:

B SIGN Ereignisse, die beim Anmelden eines Benutzers auftreten.

B B CHANGE-PW
Ereignisse, die beim Ändern des Benutzer-Passwortes auftreten.

B B B START-PU
Ereignisse, die beim Starten eines Teilprogrammlaufs oder bei der Annahme eines Dialog- oder Asynchron-Auftrags eintreten.

B END-PU Ereignisse, die das Ende eines Teilprogrammlaufs anzeigen.

B B GSSB Ereignisse, die den Zugriff auf einen globalen sekundären Speicherbereich (GSSB) anzeigen.

B
B

TLS Ereignisse, die den Zugriff auf einen Terminal-spezifischen Langzeitspeicher (TLS) anzeigen.

B
B

ULS Ereignisse, die den Zugriff auf einen Benutzer-spezifischen Langzeitspeicher (ULS) anzeigen.

B
B
B

ADM-CMD
Ereignisse, die die Ausführung eines Administrationskommandos per Kommando oder Programmschnittstelle betreffen.

SESCHA - Session-Eigenschaften für verteilte Verarbeitung über LU6.1 festlegen

Mit der Steueranweisung SESCHA werden Eigenschaften für Sessions festgelegt (Session Charakteristika zwischen der lokalen und der Partner-Anwendung). Der mit der SESCHA-Anweisung definierte Satz von Session-Eigenschaften wird unter dem Namen *sescha_name* abgelegt und kann in der LPAP-Anweisung (siehe Seite 341) einem LPAP-Partner mit dem Operanden SESCHA= zugeordnet werden.

Bitte beachten Sie bei der Generierung von LU6.1-Verbindungen die Informationen in Abschnitt „Verteilte Verarbeitung über das LU6.1-Protokoll“ auf Seite 72.

```
SESCHA␣      sescha_name
             [ ,CONNECT={ YES | NO } ]
             [ ,CONTWIN={ YES | NO } ]
             [ ,DPN=destination_process_name ]
             [ ,IDLETIME=idle_time ]
             [ ,PACCNT=pacing_count_number ]
               ,PLU={ YES | NO }
X/W          zusätzlicher Operand in Unix-, Windows-Systemen
X/W          [ ,MAP={ USER | SYSTEM } ]
```

sescha_name definiert den Namen, unter dem die Sessioneigenschaften zusammengefasst werden. Dieser Name wird in der LPAP-Anweisung beim Operanden SESCHA= angegeben, um die Session Characteristika einem LPAP-Partner zuzuordnen.

CONNECT= legt fest, ob die lokale Anwendung beim Anwendungsstart die Verbindung zur Partner-Anwendung aufbauen soll.

NO Die Verbindung zur Partner-Anwendung muss per Administrationskommando aufgebaut werden.

Standard: NO

YES Beim Start der lokalen Anwendung wird die Verbindung zur Partner-Anwendung aufgebaut.
Kommt die Verbindung beim Start nicht zustande, so wird im Abstand von MAX ...,CONRTIME= versucht, die Verbindung aufzubauen.

Lokale Anwendung und Partner-Anwendung dürfen beide CONNECT=Y angeben. Dadurch wird erreicht, dass die Verbindung automatisch aufgebaut wird, sobald beide Anwendungen verfügbar sind.

CONTWIN= (**cont**ention **win**ner)
legt fest, ob die lokale Anwendung Contention Winner oder Contention Loser ist. Die Contention-Winner-Anwendung verwaltet die Session und regelt die Belegung der Session durch Aufträge. In einer der beiden beteiligten Anwendungen ist CONTWIN=Y anzugeben und in der anderen CONTWIN=N.

YES Die Partner-Anwendung ist Contention Winner.

NO Die lokale Anwendung ist Contention Winner.

In jedem Fall können jedoch Aufträge sowohl von der lokalen als auch von der Partner-Anwendung gestartet werden. Im Konfliktfall, wenn gleichzeitig die lokale und die Partner-Anwendung einen Auftrag starten wollen, wird die Session mit dem Auftrag der Contention-Winner-Anwendung belegt.

Die korrekte Wahl dieses Parameters ist wichtig für die Performance in der Kommunikation zwischen zwei Anwendungen: In der Anwendung, von der aus am häufigsten Aufträge gestartet werden, sollte CONTWIN=N generiert werden.

Standard: Bei PLU=N wird die lokale Anwendung Contention Loser (CONTWIN=YES), bei PLU=Y Contention Winner (CONTWIN=NO).

DPN=destination_process_name
Eine asynchrone Nachricht wird immer von der bei DPN angegebenen Instanz bearbeitet. DPN= ist nur bei Kopplung zu IBM-Systemen relevant.

Standard: 8 Leerzeichen

IDLETIME=idle_time
Zeit in Sekunden zur Überwachung des Idle-Zustands einer Session. Wird während des mit IDLETIME= festgelegten Zeitraums die Session von keinem Auftrag belegt, baut openUTM die Verbindung ab.

IDLETIME=0 bewirkt, dass der Idle-Zustand nicht überwacht wird.

Standardwert: 0
Minimalwert: 0
Maximalwert: 32767

X/W X/W X/W X/W MAP= regelt die Umwandlung ASCII/EBCDIC beim Austausch unformatierter Nachrichten mit anderen Anwendungen.
Bei formatierten Nachrichten (KCMF enthält ein Formatkennzeichen) führt openUTM grundsätzlich keine Nachrichtenbehandlung durch.

X/W X/W X/W USER openUTM führt keine Nachrichtenbehandlung durch, d.h. die Daten des Nachrichtenbereichs werden unverändert an die Partner-Anwendung übertragen.

X/W Standard: USER

X/W X/W X/W SYSTEM openUTM konvertiert die Daten des Nachrichtenbereichs vor dem Absenden von ASCII nach EBCDIC bzw. nach dem Empfangen von EBCDIC nach ASCII. Die Nachricht darf dabei nur abdruckbare Zeichen enthalten.

PACCNT=pacing_count_number

Anzahl der Teil-Nachrichten, die eine lokale Anwendung bei längeren Nachrichten maximal unbeantwortet empfangen kann. Ein zu großer Wert kann zu Stauproblemen im Netz führen.

Bei PACCNT=0 findet kein pacing statt.

Standard: 3
Minimalwert: 0
Maximalwert: 63

VORSICHT!
Werden mit der Partner-Anwendung nur kurze Nachrichten ausgetauscht (kleiner 4000 Byte), dann sollte das Pacing ausgeschaltet werden (PACCNT=0); dies erspart Overhead in der Kommunikation mit dem Partner. Treten dennoch Datenflussprobleme auf, dann muss entweder der Standard wieder eingestellt oder die Generierung des Transportsystems angepasst werden.

PLU= legt fest, welche Anwendung den Aufbau der Session initiiert, d.h. welche Anwendung die primary logical unit (PLU) ist.

YES Die Partner-Anwendung ist die primary logical unit.

NO Die lokale Anwendung ist die primary logical unit.

Für eine der beteiligten Anwendungen muss PLU=Y angegeben werden und für die andere Anwendung PLU=N.

SFUNC - Funktionstasten definieren

Mit dieser Steueranweisung können den Funktionstasten von Terminals zugeordnet werden:

- Transaktionscodes,
- KDCS-Returncodes, die den Teilprogrammen übergeben werden,
- KDC-Kommandos,
- die Kellerungsfunktion.

Diese Steueranweisung ist für jede Funktionstaste, die Sie verwenden wollen, genau einmal anzugeben.

Ab openUTM-Client mit Trägersystem UPIC V4.0 kann von UPIC-Clients eine Funktionstaste ausgewählt und an die UTM-Anwendung übertragen werden.
Empfängt openUTM von einem UPIC-Client eine Funktionstaste, so wird nur der Parameter RET ausgewertet. Ist der Parameter nicht generiert, so liefert openUTM beim MGET-Aufruf den Returncode 19Z.

B B

- Mit SFUNC belegte F- und K-Tasten dürfen von FHS-DE nicht benutzt werden (siehe Benutzerhandbuch „FHS“).

X/W X/W

- Auf Unix- und Windows-Systemen sind Funktionstasten nur für UPIC-Clients relevant. Es wird nur der Parameter RET ausgewertet.

```
SFUNC␣        functionkey
              { [ ,CMD={ KDCDISP | KDCFOR[1] | KDCLAST | KDCOFF | KDCOFF-BUT |
                         KDCOUT | KDCSIGN[1] } ]
                |
                [ ,TAC=tac1 ] {[,RET=xxZ] | [,STACK=tac2]} }
```

B [1] nur in BS2000/OSD

functionkey Kurzbezeichnung der Funktionstaste. Zugelassen sind:

B X/W BS2000/OSD: K1 bis K14 und F1 bis F24
Unix-, Windows-Systeme: F1 bis F20

Mit F-Tasten können der Wert der F-Taste und eine Eingabe-Nachricht übermittelt werden.

B B B B Mit K-Tasten werden Kurz-Nachrichten ausgelöst, bei denen nur der Wert der K-Taste übermittelt wird.
K14 wird für Ausweisleser benötigt (siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“, Ausweisleser).

CMD= bezeichnet den Namen eines KDC-Kommandos, das dieser Funktionstaste zugeordnet werden soll. Wird der Operand CMD angegeben, dann können keine weiteren Operanden angegeben werden.

TAC=tac1 Name des Transaktionscodes, der dieser Funktionstaste zugeordnet sein soll. Der Transaktionscode muss mit einer TAC-Anweisung als Vorgangs-TAC definiert werden (CALL=FIRST/BOTH). Das Drücken der Funktionstaste ist gleichbedeutend mit der Eingabe des Transaktionscodes, falls der Benutzer sich außerhalb eines Vorgangs befindet. Drückt er diese Funktionstaste innerhalb eines Vorgangs und ist weder RET noch STACK angegeben, dann liefert der erste MGET-Aufruf im nächsten Teilprogramm dieses Vorgangs den Returncode 19Z.

RET=xxZ Returncode, der im Feld KCRCCC des Kommunikationsbereiches nach dem MGET-Aufruf steht, wenn die Funktionstaste vom Benutzer während eines Vorgangs gedrückt wurde.

Wird die Funktionstaste zu Vorgangsbeginn am Terminal gedrückt und ist TAC=*tac1* nicht gesetzt, dann reagiert openUTM mit der Meldung K009 oder startet das BADTACS-Teilprogramm.
Das BADTACS-Teilprogramm erhält beim ersten MGET-Aufruf den der Funktionstaste zugeordneten Returncode im Feld KCRCCC.

Wird die Funktionstaste bei Beginn einer Conversation vom UPIC-Client ausgelöst, so wird der Vorgang gestartet, der zu dem vom UPIC-Client gesetzten TAC (TP_NAME) gehört. Das Teilprogramm erhält den der Funktionstaste zugeordneten Returncode beim ersten MGET-Aufruf im Feld KCRCCC.

Die Operanden RET und STACK dürfen nicht zusammen angegeben werden.

Werte: $20 \leq xx \leq 39$.
Die Zuordnung kann beliebig gewählt werden.

STACK=tac2 Name eines Transaktionscodes, der dieser Funktionstaste zugeordnet werden soll. Der Transaktionscode muss mit einer TAC-Anweisung als TAC eines Dialog-Vorgangs definiert werden (TYPE=D und CALL=FIRST/BOTH). Befindet sich der Benutzer innerhalb eines Vorgangs, dann bewirkt das Drücken der Funktionstaste eine Kellerung des aktuellen Vorgangs und das Starten des Vorgangs mit dem Transaktionscode *tac2*. Befindet sich der Benutzer außerhalb eines Vorgangs, wird der Transkationscode *tac1* gestartet. Ist der Transkaktionscode *tac1* nicht angegeben, dann bewirkt das Drücken der Funktionstaste das Starten des Vorgangs mit dem Transaktionscode *tac2*.

Die Operanden RET= und STACK= dürfen nicht zusammen angegeben werden.

B **Ersatzbelegung unter BS2000/OSD**

B Die folgenden Ersatzbelegungen für K- und F-Tasten, die auf der Tastatur nicht vorhanden
B sind, können angegeben werden:

Taste	Ersatz
K1	
K2	
K3	
K4	ESC V
K5	ESC W
K6	ESC M
K7	ESC N
K8	ESC O
K9	ESC ?
K10	ESC >
K11	ESC =
K12	ESC <
K13	ESC ;
K14	ESC :
F1	
F2	
F3	
F4	ESC ^
F5	ESC _

B Das 9763-Terminal verfügt über 24 Funktionstasten (F1 bis F24). Dabei sind die Tasten
B F1 bis F20 mit **SHIFT und** entsprechender Taste Fx(x) sowie die Tasten F21 bis F24 mit
B **CTRL und** entsprechender F-Taste zu erreichen.

SHARED-OBJECT - Shared Objects/DLLs definieren (Unix-, Windows-Systeme)

X/W Mit der SHARED-OBJECT-Anweisung werden festgelegt:

X X
- unter Unix-Systemen: Namen und Eigenschaften von Shared Objects, wenn beim Programmaustausch mit dem dynamischen Binder gearbeitet werden soll.

W W
- unter Windows-Systemen: Namen und Eigenschaften dynamisch nachzuladender DLLs.

X X Die Funktionen für den Programmaustausch sind auf allen Unix-Systemen außer auf AIX-Systemen verfügbar.

```
X/W  SHARED-OBJECT␣  shared_object_name
X/W                  [ ,DIRECTORY=directory_name ]
X/W                  [ ,LOAD-MODE={ STARTUP | ONCALL } ]
X/W                  [ ,VERSION=version ]
```

X/W shared_object_name
X/W X/W Name des Shared Objects/DLLs. *shared_ object_ name* kann max. 32 Zeichen lang sein.

X/W DIRECTORY= *directory_name* bezeichnet das Verzeichnis, in dem das Shared Object/DLL
X/W abgelegt ist. *directory_name* kann max. 54 Zeichen lang sein.

X/W Standard: keine Angabe, d.h. es wird das aktuelle Verzeichnis benutzt

X *Unix-Systeme:*
X Wird mit DIRECTORY= kein Verzeichnis angegeben, wird das Shared
X Object zunächst im Verzeichnis *filebase* gesucht. Falls dies misslingt, wird
X als Suchpfad die Umgebungsvariable LD_LIBRARY_PATH benutzt.

X/W LOAD-MODE=
X/W Lademodus des Shared Objects/DLL.

X/W STARTUP Das Shared Object/DLL wird beim Start der Anwendung geladen.

X/W Standard: STARTUP

X/W ONCALL Das Shared Object/DLL wird beim erstmaligen Aufruf eines Teilprogramms
X/W oder VORGANG-Exits geladen.

X/W X/W X/W Shared Objects, die mit LOAD-MODE=ONCALL generiert sind, können nur ausgetauscht werden, wenn die openUTM-Versionsunterstützung für Shared Objects/DLLs verwendet wird.

X/W X/W VERSION= Versionsbezeichner des Shared Objects/DLLs. *version* kann max. 24 Zeichen lang sein.

X/W X/W X/W X/W VERSION= wird nur dann bewertet, wenn die Versionsunterstützung durch openUTM benutzt wird. Wie die Versionsunterstützung eingesetzt wird, ist im openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter Unix- und Windows-Systemen“ beschrieben.

X/W Standard: keine Versionsangabe

W W W W **i** Für Anwendungen unter Windows-Systemen wird dringend empfohlen, den Operanden VERSION= zu verwenden, da die Suche nach dem „lexikalisch größten Namen“ auf Windows-Systemen unerwartete Ergebnisse liefern kann.

X/W X/W X/W X/W X/W X/W Wird VERSION= nicht angegeben und ist *shared_object_name* ein Verzeichnisname, so wird das Shared Object mit dem lexikalisch größten Namen angesprochen. Für openUTM ist der Versionsbezeichner lediglich ein Identifikator, die lexikalische Ordnung bedeutet beispielsweise nicht „älter“ oder „neuer“. Für die Versionsverwaltung ist der UTM-Administrator verantwortlich.

X/W Shared Object-Dateiname **ohne** Versionsunterstützung:

X X X X X *Unix-Systeme*:
Der vollständige Dateiname des Shared Objects setzt sich zusammen aus *directory_name/shared_object_name*. Ist *directory_name/shared_object_name* ein Verzeichnis, so wird die Datei mit dem lexikalischen größten Dateinamen aus diesem Verzeichnis geladen.

X X Z.B. für die Generierungsangaben: `SHARED-OBJECT aaa.so, DIRECTORY=.` wird die Datei `./aaa.so` geladen.

X/W Shared Object-Dateiname **mit** Versionsunterstützung:

X
X
X *Unix-Systeme*:
Der vollständige Dateiname des Shared Objects setzt sich zusammen aus *directory_name/shared_object_name/version.*

X
X
X Für die Generierungsangaben
`SHARED-OBJECT aaa, DIRECTORY=., VERSION=V1.SO`
wird die Datei `./aaa/V1.SO` geladen.

W
W
W *Windows-Systeme:*
Der vollständige Dateiname des Shared Objects setzt sich zusammen aus *directory_name\shared_object_name\version.*

W
W
W Für die Generierungsangaben
`SHARED-OBJECT aaa, DIRECTORY=., VERSION=V1`
wird die Datei `.\aaa\V1.dll` geladen.

W Das Suffix `.dll` wird automatisch angehängt.

SIGNON - Anmeldeverfahren steuern

Mit der Steueranweisung SIGNON können Sie Optionen und Parameter für das Anmeldeverfahren Ihrer UTM-Anwendung festlegen. Durch die SIGNON-Parameter wird die Anmeldung von Benutzern gesteuert.

Die Parameter UPIC, RESTRICTED und CONCURRENT-TERMINAL-SIGNON sind nur relevant, wenn ein Anmelde-Vorgang generiert ist.

Geben Sie für die Operanden von SIGNON ungültige Werte ein, dann setzt KDCDEF den jeweiligen Standardwert. Dies geschieht z.T. ohne Ausgabe einer entsprechenden Meldung (siehe folgende Operandenbeschreibung).

```
SIGNON␣      [ CONCURRENT-TERMINAL-SIGNON=%_value ]
             [ ,GRACE={ NO | YES } ]
             [ ,MULTI-SIGNON={ YES | NO } ]
             [ ,OMIT-UPIC-SIGNOFF={ YES | NO } ]
             [ ,PW-HISTORY=number ]
             [ ,RESTRICTED={ YES | NO } ]
             [ ,SILENT-ALARM=number1 ]
             [ ,UPIC={ YES | NO } ]
```

CONCURRENT-TERMINAL-SIGNON=%_value

ist nur relevant, wenn in Ihrer Anwendung ein Anmelde-Vorgang generiert ist.
In CONCURRENT-TERMINAL-SIGNON geben Sie an, für wieviel Prozent der generierten Benutzer gleichzeitig ein Anmelde-Vorgang aktiv sein kann. openUTM versucht, entsprechend dieser Angabe, die notwendigen Betriebsmittel anzulegen.

Der Wert *%_value* bezieht sich nur auf Anmelde-Vorgänge, die für Terminal-Benutzer und TS-Anwendungen gestartet werden.

Standard: 25 (%)
Minimalwert: 1 (%)
Maximalwert: 100 (%)

Wenn Sie für *%_value* einen Wert < 1 oder > 100 angeben, setzt KDCDEF ohne Meldung den Standardwert von 25 % ein.

GRACE= (Grace-Sign-On)
legt fest, ob ein Benutzer nach Ablauf der Passwort-Gültigkeit (siehe USER PROTECT-PW, Seite 546) sein Passwort noch ändern darf.

YES Der Benutzer kann sein Passwort nach Ablauf der Gültigkeit noch ändern. Die Änderung muss innerhalb des Anmeldeverfahrens erfolgen, bevor der Benutzer vollständig angemeldet ist.
Falls ein Anmelde-Vorgang aktiviert ist, dann kann das Passwort dort mit dem KDCS-Aufruf SIGN CP geändert werden, unabhängig vom Client-Typ. Ein Anmelde-Vorgang wird immer aktiviert, wenn sich ein Benutzer über eine Verbindung anmeldet, für deren Transportzugangspunkt ein Anmelde-Vorgang generiert ist.

Die folgende Tabelle zeigt, wie sich die einzelnen Client-Typen bei abgelaufenem Passwort verhalten und wie das Verhalten davon abhängt, ob ein Anmelde-Vorgang aktiviert ist.

	Client-Typ	**Verhalten, wenn das Passwort abgelaufen ist**[1]
	UPIC ab V5.1	Das Passwort kann unabhängig von der Aktivierung eines Anmelde-Vorgangs auch mit der Funktion *Set_Conversation_Security_New_Password* geändert werden.
	UPIC bis V5.0	Der Benutzer kann das Passwort ohne Aktivierung eines Anmelde-Vorgangs nicht mehr ändern.
B B B	BS2000-Terminal	Bei dunkelgesteuertem Passwort fordert openUTM den Benutzer zur Änderung des Passwortes auf, unabhängig davon, ob ein Anmelde-Vorgang aktiviert ist.
B B B		Bei nicht-dunkelgesteuertem Passwort fordert openUTM den Benutzer nur dann zur Änderung des Passwortes auf, wenn kein Anmelde-Vorgang aktiviert ist.
X/W X/W X/W	Terminal auf Unix-/Windows-Systemen	openUTM fordert den Benutzer zur Änderung des Passwortes auf, unabhängig davon, ob ein Anmelde-Vorgang aktiviert ist.
	TS-Anwendung	Der Benutzer kann das Passwort ohne Aktivierung eines Anmelde-Vorgangs nicht mehr ändern.

[1] Das Passwort kann immer über die Administrations-Schnittstelle geändert werden. Standardmäßig werden Passwörter mit beschränkter Gültigkeitsdauer bei Änderung über die Administrations-Schnittstelle sofort auf „abgelaufen“ gesetzt. Wollen Sie dies verhindern, müssen Sie dies an der Administrations-Schnittstelle explizit anfordern.

Nach einer Neu- oder Änderungsgenerierung sind folgende Besonderheiten zu beachten:

– Wird das Passwort eines Benutzers nach einer Neugenerierung (mit anschließendem KDCUPD-Lauf) auf Grund der Erhöhung der Komplexität ungültig, so kann der Benutzer sein Passwort nur im Anmelde-Vorgang (mit SIGN CP) ändern.

– Nach einer Neugenerierung (ohne nachfolgenden KDCUPD-Lauf) erzwingt openUTM beim ersten Anmelden die Änderung von Passwörtern, die mit einer Gültigkeitsdauer generiert sind.

NO Der Benutzer kann sein Passwort nach Ablauf der Gültigkeit nicht mehr ändern. Das Passwort kann nach Ablauf der Gültigkeit nur noch vom Administrator geändert werden.

Standard: NO

MULTI-SIGNON=

legt fest, ob sich ein Benutzer gleichzeitig mehrfach unter derselben Benutzerkennung an die Anwendung anmelden kann.

Der Operand MULTI-SIGNON hat keine Auswirkung auf das Empfangen und Starten von Asynchron-Vorgängen über OSI TP.

YES Folgende Fälle sind zu unterscheiden:

– Die Benutzerkennung ist mit USER..,RESTART=NO generiert:
In diesem Fall darf sich ein Benutzer mehrfach gleichzeitig mit dieser Benutzerkennung bei der Anwendung anmelden. Dabei darf jedoch nur eine Anmeldung über ein Terminal erfolgen. Über UPIC-, APPLI-, SOCKET- und OSI TP-Verbindungen können zusätzlich weitere Anmeldungen erfolgen.

– Die Benutzerkennung ist mit USER...,RESTART=YES generiert:
In diesem Fall darf sich ein Benutzer höchstens einmal an die Anwendung anmelden, jedoch können in der Anwendung für den Benutzer zusätzlich mehrere Auftragnehmer-Vorgänge aktiv sein, sofern diese über OSI TP-Verbindungen gestartet und die Functional Unit „Commit“ ausgewählt wurde.

NO Jede Benutzerkennung darf gleichzeitig höchstens einmal angemeldet sein und für jede Benutzerkennung kann maximal ein Dialog-Vorgang gleichzeitig aktiv sein.

Standard: YES

OMIT-UPIC-SIGNOFF=

legt fest, ob ein Benutzer, der sich über eine UPIC-Verbindung angemeldet hat, nach Ende der Conversation angemeldet bleibt oder nicht.

YES Wenn sich ein Benutzer über eine UPIC-Verbindung angemeldet hat, dann bleibt er nach Ende der Conversation angemeldet. Er wird erst abgemeldet,
- wenn vor dem Start einer neuen UPIC Conversation über dieselbe Verbindung im UPIC-Protokoll ein anderer User übergeben wird,
- oder wenn die Verbindung abgebaut wird.

Wenn im UPIC-Protokoll kein anderer User übergeben wird, wird vor dem Start der UPIC Conversation kein Anmeldevorgang gestartet.

Wenn die Anwendung ohne User generiert ist, kommt es für eine bestehende Verbindung nie zu einem Wechsel der UserId. Deshalb wird in diesem Fall nur vor dem Start der ersten Conversation nach dem Verbindungsaufbau gegebenenfalls ein Anmeldevorgang gestartet.

Standard in UTM-Cluster-Anwendungen.

NO Wenn sich ein Benutzer über eine UPIC-Verbindung angemeldet hat, wird er am Ende der UPIC-Conversation abgemeldet.

Standard in stand-alone Anwendungen.

PW-HISTORY=number

legt fest, ob und über wieviele Passwort-Änderungen openUTM eine Passwort-Historie führen soll.

Geben Sie für *number* einen Wert > 0 an, dann führt openUTM eine Passwort-Historie. *number* ist die Anzahl der Passwörter einer Benutzerkennung, die openUTM sich merkt.

Ändert ein Benutzer sein Passwort und ist für das Passwort in der USER-Anweisung eine maximale Gültigkeitsdauer generiert, dann muss das neue Passwort sich von dem aktuellen und den letzten *number* Passwörtern, die für die Benutzerkennung gesetzt wurden, unterscheiden.
number=0 bedeutet, dass openUTM keine Passwort-Historie führt.

Standard: 0
Minimalwert: 0
Maximalwert: 10

Geben Sie für PW-HISTORY einen Wert > 10 an, dann setzt KDCDEF den Maximalwert 10 ein.

Die Passwort-Historie gilt nur für den Benutzer, eine Änderung des Passworts per Administration ist unabhängig von der Historie möglich.

RESTRICTED=

legt fest, ob im 1. Teil des Anmelde-Vorgangs DB-Aufrufe und Zugriffe auf globale UTM-Speicher verboten sind.

YES Im 1. Teil des Anmelde-Vorgangs sind DB-Aufrufe und Zugriffe auf globale UTM-Speicher verboten.

NO Im 1. Teil des Anmelde-Vorgangs sind DB-Aufrufe und Zugriffe auf globale UTM-Speicher erlaubt.

Standard: YES

SILENT-ALARM=number1

legt die Anzahl der aufeinander folgenden erfolglosen Anmeldeversuche über einen LTERM-Partner oder von einem Benutzer fest, nach der ein stiller Alarm (K094-Meldung) ausgelöst wird. Die Meldung wird nach *number1* aufeinander folgenden ungültigen Anmeldeversuchen eines Benutzers oder von einem Client aus ausgegeben.

Standard: 10
Minimalwert: 1
Maximalwert: 100

UPIC= ist nur relevant, wenn in Ihrer Anwendung ein Anmelde-Vorgang generiert ist.
Mit UPIC= legen Sie fest, ob der Anmelde-Vorgang aktiviert wird, wenn ein UPIC-Client eine Conversation starten will.

YES Wenn für den Transportsystemendpunkt (BCAMAPPL), über den sich der UPIC-Client mit der Anwendung verbunden hat, ein Anmeldevorgang generiert ist, wird dieser vor jeder UPIC Conversation gestartet.

NO Für einen UPIC-Client wird kein Anmelde-Vorgang gestartet.

Standard: NO

TAC - Eigenschaften von Transaktionscodes und TAC-Queues festlegen

Mit der Steueranweisung TAC werden Namen und Eigenschaften von Transaktionscodes und TAC-Queues der UTM-Anwendung festgelegt.

Transaktionscodes sind dabei „Rufnamen“ für Teilprogramme der Anwendung. Einem Transaktionscode müssen Sie immer einen Programmnamen (Operand PROGRAM=) zuweisen.

TAC-Queues sind anwendungsweite Message-Queues, die unabhängig von einem Teilprogramm existieren. Daher darf der Operand PROGRAM= nicht angegeben werden. TAC-Queues sind Service-gesteuert, d.h. für das Lesen von Nachrichten aus Queues sind die Teilprogramme der UTM-Anwendung verantwortlich, openUTM übernimmt - im Gegensatz zu Transaktionscodes - kein Scheduling.

Die **Dead Letter Queue** ist eine TAC-Queue mit dem festen Namen KDCDLETQ. Sie steht immer zur Verfügung, um Asynchron-Nachrichten an Transaktionscodes oder TAC-Queues zu sichern, die endgültig, d.h. nach evtl. erfolgter maximaler Anzahl erneuter Zustellungen, nicht verarbeitet werden konnten. Diese Nachrichten können mit DGET BF/BN gelesen und mit DADM MV/MA zur weiteren Verarbeitung in andere Message Queues verschoben werden. Für die Dead Letter Queue KDCDLETQ können Sie keine Nachrichten erzeugen oder verarbeiten.

Die Sicherung von Asynchron-Nachrichten in der Dead Letter Queue kann durch den Parameter DEAD-LETTER-Q der TAC-Anweisung für jedes Nachrichtenziel einzeln ein- und ausgeschaltet werden. Hauptaufträge zu Message-Komplexen mit negativen Quittungsaufträgen werden nie in der Dead Letter Queue gesichert.

Für die Dead Letter Queue wird bei der Generierung der Name KDCDLETQ erzeugt. Dabei werden folgende Eigenschaften gesetzt:

TYPE=Q, STATUS=ON, ADMIN=N, QMODE=STD, QLEV=32767

Die Eigenschaften dieser TAC-Queue können auch durch eine eigene TAC-Anweisung definiert werden.

Eine Nachricht einer TAC-Queue kann nicht verarbeitet werden, wenn die Transaktion zurückgesetzt wird, die den Aufruf DGET FT/NT oder PF/PN beinhaltet. Eine Nachricht an einen Asynchron-TAC kann nicht verarbeitet werden, wenn der gestartete Asynchron-Vorgang mit PEND ER/FR abnormal beendet wird, ohne vorher einen Sicherungspunkt erreicht zu haben.

Generieren von Transaktionscodes

- Die Parameter QMODE, Q-READ-ACL und Q-WRITE-ACL sind für Transaktionscodes ohne Bedeutung.
- Bei der Definition von Transaktionscodes für Teilprogramme, die Aufrufe der X/Open-Schnittstellen CPI-C oder XATMI enthalten, müssen Sie dem TAC mit dem Operanden API= das Kennzeichen für die verwendete Programmschnittstelle zuordnen.
- Administrationskommandos, die Sie zur Administration der Anwendung nutzen wollen, müssen Sie ebenfalls als TAC definieren. Administrationskommandos können Sie als Dialog-TACs und als Asynchron-TACs generieren. Es muss mindestens ein Administrations-TAC (vorzugsweise das Administrationkommando KDCSHUT) generiert und in der Anwendung definiert sein. Außerdem muss mindestens ein Benutzer mit Administrationsberechtigung generiert werden.
- Die Event-Services BADTACS, MSGTAC werden dadurch definiert, dass man TAC-Anweisungen mit den privilegierten TAC-Namen KDCBADTC und KDCMSGTC in die Generierung aufnimmt.
- Ein Event-Service SIGNON (= Anmelde-Vorgang) kann auf mehrere Arten definiert werden:
  - durch den privilegierten TAC-Namen KDCSGNTC. Damit definieren Sie den Event-Service für den in MAX APPLINAME=*appliname* angegebenen Zugangspunkt. Dieser Event-Service ist dann zugleich Standard für alle anderen Zugangspunkte, die mit einer BCAMAPPL-Anweisung generiert werden.
  - durch BCAMAPPL *appliname2*,SIGNON-TAC=*signon-tac* zusammen mit TAC *signon-tac*. Damit definieren Sie einen eigenen Event-Service für den Zugangspunkt *appliname2*. Auf diese Weise können Sie mehrere SIGNON-Services definieren.

  Der mit KDCSGNTC generierte Event-Service ist Standard für alle anderen Zugangspunkte, die mit einer BCAMAPPL-Anweisung generiert werden.

- Für die Event-Services BADTACS, MSGTAC und SIGNON gelten für einige Operanden Voreinstellungen, die in unten stehender Tabelle aufgelistet sind. Diese Voreinstellungen können bei KDCBADTC, KDCMSGTC und KDCSGNTC nicht verändert werden. Bei TAC *signon-tac* heißt das, dass Sie die Werte so setzen müssen.

Operand in TAC-Anweisung	**Voreinstellung bei**		
	KDCBADTC	**KDCMSGTC**	**KDCSGNTC bzw. TAC signon-tac**
ACCESS-LIST=	Leerzeichen	Leerzeichen	Leerzeichen
ADMIN=	NO	(frei wählbar)	NO
API=	KDCS	KDCS	KDCS
CALL=	FIRST	FIRST	BOTH
ENCRYPTION-LEVEL=	NONE	NONE	NONE
LOCK=	0	0	0
B B SATADM= (BS2000/OSD)	NO	NO	NO
B B SATSEL= (BS2000/OSD)	NONE	NONE	NONE
STATUS=	OFF	OFF	OFF
TACCLASS=	keine TAC-Klasse	16	keine TAC-Klasse
TYPE=	D	A	D

Diese Voreinstellungen bedeuten z.B., dass die Transaktionscodes KDCSGNTC, KDCBADTC und KDCMSGTC keinem Zugriffsschutz durch Keysets und Lockcodes unterliegen und weder vom Benutzer noch in einem FPUT- bzw. DPUT-Aufruf verwendet werden können.

Außerdem unterliegen die TACs KDCBADTC, KDCSGNTC und KDCMSGTC nicht der Bearbeitungssteuerung über die TAC-Klassen. Das gilt auch für KDCMSGTC, obwohl KDCMSGTC der TAC-Klasse 16 zugeordnet ist.

Weiterhin unterliegen alle TACs, die innerhalb eines Anmelde-Vorgangs laufen, nicht der Bearbeitungssteuerung über die TAC-Klassen.

- Für KDCMSGTC ist DEAD-LETTER-Q=NO fest eingestellt und nicht änderbar.

B
- Beachten Sie bei der Generierung der TACs Folgendes:

B
B
B
B
  - Die den TACs KDCBADTC, KDCMSGTC und KDCSGNTC und TAC *signon-tac* zugeordneten Programme dürfen keinem Lademodul zugeordnet werden, das erst beim ersten Aufruf eines seiner Teilprogramme nachgeladen werden soll (LOAD-MODULE-Anweisung mit LOAD-MODE=ONCALL).

B
B
B
  - Der Event-Exit VORGANG und die Teilprogramme des Vorgangs müssen im gleichen Lademodul liegen, wenn das Lademodul mit LOAD-MODE=ONCALL generiert wird.

B
B
B
- UTM-SAT-Administrationskommandos (Preselection-Kommandos) können nur als Dialog-TACs generiert werden. Die Namen dieser TACs finden Sie im openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“.

Generieren von TAC-Queues

Für die Generierung einer TAC-Queue (TYPE=Q) sind nur die folgenden Operanden der TAC-Anweisung von Bedeutung:

tacname, ADMIN, DEAD-LETTER-Q, QLEV, QMODE, Q-READ-ACL, Q-WRITE-ACL, STATUS und TYPE.

Für die Dead Letter Queue KDCDLETQ können die Operanden ADMIN, QLEV, QMODE, Q-READ-ACL und STATUS frei verwendet werden.

Alle anderen Operanden werden für TAC-Queues nicht ausgewertet.

Näheres zu TAC-Queues und den damit möglichen Anwendungen finden Sie im openUTM-Handbuch „Konzepte und Funktionen“.

```
TAC␣        tacname
            [ ,{ ACCESS-LIST=keysetname | LOCK=lockcode } ]
            [ ,ADMIN={ YES | NO | READ } ]
            [ ,API={ KDCS | ( XOPEN,{ XATMI | CPIC } ) ]
            [ ,CALL={ BOTH | FIRST | NEXT } ]
            [, DEAD-LETTER-Q={ NO | YES }
            [ ,ENCRYPTION-LEVEL={ NONE | 1 | 2 } ]
            [ ,EXIT=conversation_exit ]
            [ ,PGWT={ NO | YES } ]      nur bei Verwendung von TAC-PRIORITIES erlaubt
            [ ,PROGRAM=objectname ]     nur mit TYPE=D | A erlaubt
            [ ,QLEV=queue_level_number ]
            [ ,QMODE = { STD | WRAP-AROUND } ]
            [ ,Q-READ-ACL = read-keysetname ]
            [ ,Q-WRITE-ACL = write-keysetname ]
            [ ,STATUS={ ON | OFF | HALT | KEEP } ]
            [ ,TACCLASS=tacclass ]
            [ ,TACUNIT=tacunit ]
            [ ,TYPE={ D | A | Q } ]

B           zusätzliche Operanden in BS2000/OSD
B           [ ,DBKEY=dbkey ]
B           [ ,RUNPRIO=priority ]
B           [ ,SATADM={ NO | YES } ]
B           [ ,SATSEL={ BOTH | SUCC | FAIL | NONE } ]
B           [ ,TCBENTRY=name_of_tcbentry_statement ]
B           [ ,TIME={ time1 | (time1,time2) } ]

X/W         zusätzlicher Operand in Unix- und Windows-Systemen
X/W         [ ,RTIME=rtime ]
```

tacname Max. 8 Zeichen langer Name des Transaktionscodes bzw. der TAC-Queue.

tacname muss eindeutig sein und darf keinem weiteren Objekt der Namensklasse 1 zugeordnet sein. Siehe dazu auch Abschnitt „Eindeutigkeit der Namen und Adressen“ auf Seite 262.

ACCESS-LIST=keysetname

Hiermit definieren Sie die Zugriffsrechte von Benutzern für diesen Transaktionscode. ACCESS-LIST= darf nicht zusammen mit dem Operanden LOCK=*lockcode* angegeben werden.

Für *keysetname* müssen Sie den Namen eines Keysets angeben. Das Keyset muss mit einer KSET-Anweisung definiert werden.

Ein Benutzer kann nur dann auf den Transaktionscode zugreifen, wenn das Keyset des Benutzers (USER ...,KSET=), das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, und das in *keysetname* angegebene Keyset mindestens einen gemeinsamen Keycode enthalten.

Geben Sie weder ACCESS-LIST=*keysetname* noch LOCK=*lockcode* an, dann ist der Transaktionscode nicht geschützt und jeder Benutzer kann den Transaktionscode aufrufen.

Standard: kein Keyset

ADMIN= gibt an, welche Berechtigung der Benutzer benötigt, um diesen Transakionscode oder diese TAC-Queue aufrufen zu können, bzw. einen Vorgang, der diesen Transaktionscode als Folge-TAC enthält.

YES Bedeutung für einen TAC (TYPE=A oder D):
Diesen TAC kann nur der Administrator bzw. ein Benutzer mit Administrationsberechtigung aufrufen. In dem zugehörigen Administrationsprogramm können alle Funktionen der Programmschnittstelle für die Administration genutzt werden.

Bedeutung für eine TAC-Queue (TYPE=Q):
Nur der Administrator bzw. ein Benutzer mit Administrationsberechtigung darf Nachrichten aus dieser Queue lesen bzw. in diese Queue schreiben.

NO Für diesen TAC bzw. diese TAC-Queue ist keine Administrationsberechtigung nötig.

READ Für diesen TAC bzw. diese TAC-Queue ist keine Administrationsberechtigung nötig.

In dem zugehörigen Administrationsprogramm können ausschließlich die Funktionen der Programmschnittstelle für die Administration genutzt werden, die lesend auf die Anwendungsdaten zugreifen (nur KDCADMI mit Operationscode KC_GET_OBJECT).

API= Programmschnittstelle, die das zum Transaktionscode gehörende Teilprogramm verwendet.

Pflichtoperand, wenn eine der X/Open-Schnittstellen CPI-C oder XATMI verwendet wird.

KDCS Das Teilprogramm ist ein KDCS-Programm.

Standard: KDCS

(XOPEN,CPIC)
Das Teilprogramm ist ein CPI-C-Programm.

(XOPEN,XATMI)
Das Teilprogramm ist ein XATMI-Programm.

CALL= gibt an, ob mit dem Transaktionscode ein Service gestartet wird, d.h. erster TAC eines Vorgangs, oder ob er Folge-TAC in einem Vorgang ist.

BOTH Der TAC kann erster TAC oder Folge-TAC in einem Vorgang sein.

Standard: BOTH

FIRST Der TAC kann nur erster TAC in einem Vorgang sein.

NEXT Der TAC kann nur Folge-TAC in einem Vorgang sein. Er kann nur mit STATUS=HALT gesperrt werden. Es können keine Asynchron-Aufträge an diesen TAC erzeugt werden.

CPI-C-Programme müssen mit CALL=FIRST oder BOTH generiert werden. XATMI-Programme müssen mit CALL=FIRST generiert werden.

B DBKEY=dbkey

B ist nur relevant, wenn das Teilprogramm Datenbankaufrufe absetzt.

B *dbkey* ist ein maximal 8 Zeichen langer Name, unter dem die Aktivitäten dieses Transaktionscodes beim Datenbanksystem registriert werden. Das Format des Schlüssels ist abhängig vom verwendeten Datenbanksystem. Der DBKEY wird derzeit nur bei UDS-Datenbanken genutzt. Ab UDS/SQL V1.2 dient der DBKEY als gesonderte Kennzeichnung der Aktivität im UDS-Monitor ("Programmname"). Bei Vorgängerversionen von openUTM wurde dieser Name auch im Rahmen der Rechteprüfung genutzt (daher auch die Bezeichnung DBKEY). Siehe dazu das Kapitel „BPRIVACY“ im UDS/SQL-Handbuch „Aufbauen und Umstrukturieren“.

B Standard: UTM

B Der Standardwert DBKEY=UTM bewirkt, dass der Wert des Startparameters DBKEY an der Schnittstelle zur Datenbank übergeben wird (siehe openUTM-Handbuch „Einsatz von openUTM-Anwendungen“, Startparameter).

DEAD-LETTER-Q=

gibt an, ob Asynchron-Nachrichten dieser Message Queue nach einer nicht ordnungsgemäßen Verarbeitung und nicht erfolgter Redelivery in der Dead Letter Queue gesichert werden sollen.

Die Überwachung der Anzahl Nachrichten in der Dead Letter Queue wird mit der Anweisung MAX ...,DEAD-LETTER-Q-ALARM ein- bzw. ausgeschaltet.

YES Nachrichten an diesen Asynchron-TAC oder diese TAC-Queue, die nicht verarbeitet werden konnten, werden in der Dead Letter Queue gesichert, sofern sie nicht erneut zugestellt werden (Redelivery) und (bei Message-Komplexen) kein negativer Quittungsauftrag definiert wurde.

NO Nachrichten an diesen Asynchron-TAC oder diese TAC-Queue, die nicht verarbeitet werden konnten, werden nicht in der Dead Letter Queue gespeichert.
Dieser Wert muss für alle Dialog-TACs und für Asynchron-TACs mit CALL=NEXT, sowie für KDCMSGTC und KDCDLETQ angegeben werden.

Standard: NO

Hauptaufträge zu Message-Komplexen (MCOM) mit negativen Quittungsaufträgen werden nie in der Dead Letter Queue gesichert, da im Fehlerfall die negativen Quittungsaufträge aktiviert werden.

Wird die Anzahl der Nachrichten der Dead Letter Queue mit QLEV beschränkt, können im Fehlerfall Nachrichten von Asynchron-TACs oder TAC-Queues verloren gehen. Wird auf die Beschränkung der Anzahl verzichtet, muss der Pagepool von openUTM ausreichend groß generiert werden. Droht ein Pagepool-Engpass, kann die Dead Letter Queue im laufenden Betrieb mit STATUS=OFF gesperrt werden.

ENCRYPTION-LEVEL=

legt die minimale Verschlüsselungsebene fest, die für einen Vorgang eingehalten werden muss, der über diesen Transaktionscode gestartet wird. Die hier festgelegte Verschlüsselungsebene gilt für alle Nachrichten, die in dem Vorgang gesendet und empfangen werden.

NONE Eine Nachrichtenverschlüsselung ist nicht erforderlich.
Für Transaktionscodes, die mit CALL=NEXT generiert werden, müssen Sie ENCRYPTION-LEVEL=NONE setzen.

Standard: NONE

1 | 2 Mit diesem Transaktionscode kann nur dann ein Vorgang gestartet werden, wenn die Eingabe-Nachricht vom Client verschlüsselt übertragen wird. Dialog-Ausgabe-Nachrichten des Vorgangs werden verschlüsselt an den Client übertragen.

Der Wert (1 oder 2) bestimmt den Algorithmus, der zur Verschlüsselung verwendet werden soll:

1 Verschlüsselung der Ein-/Ausgabe-Nachrichten nach dem DES-Algorithmus.

2 Verschlüsselung der Ein-/Ausgabe-Nachrichten nach dem AES-Algorithmus.

Verschlüsselt ein Client die erste Eingabe-Nachricht nicht mindestens mit der geforderten Verschlüsselungsebene oder unterstützt er keine Verschlüsselung, so wird der Vorgang nicht gestartet. Dabei gelten folgende Ausnahmen:
– Der aufrufende Client ist als vertrauenswürdig („trusted“ Client) generiert (PTERM/TPOOL...,ENCRYPTION-LEVEL=TRUSTED).
– Der Vorgang ist ein Asynchron-Vorgang und wird lokal gestartet.
– Der Vorgang wird durch Vorgangskettung gestartet.
– Der Vorgang wird ohne Benutzerdaten gestartet.

Wird der Transaktionscode durch Vorgangskettung gestartet, dann muss die erste Eingabe-Nachricht des Client nicht verschlüsselt sein.

Wird der Transaktionscode ohne Benutzerdaten aufgerufen oder durch Vorgangskettung gestartet, dann muss der Client die Fähigkeit zur Verschlüsselung haben, da openUTM alle Dialog-Ausgabe-Nachrichten verschlüsselt überträgt, und, bei Mehrschritt-Vorgängen, alle weiteren Eingabe-Nachrichten vom „not-trusted“ Client verschlüsselt erwartet.

ENCRYPTION-LEVEL=1 | 2 dürfen Sie nur für Transaktionscodes angeben, mit denen ein Vorgang gestartet wird (CALL=FIRST oder CALL=BOTH).

In Systemen, in denen die Verschlüsselungsfunktionalität von openUTM nicht installiert ist, können Transaktionscodes mit ENCRYPTION-LEVEL=1 | 2 nur von Clients gestartet werden, die als vertrauenswürdig (trusted) generiert sind.

EXIT=conversation_exit

Name des Event-Exits VORGANG, der diesem TAC zugeordnet werden soll. EXIT= darf nur zusammen mit CALL=FIRST oder CALL=BOTH angegeben werden. Der Event-Exit VORGANG muss mit einer eigenen PROGRAM-Anweisung definiert werden.

Standard: Kein Event-Exit VORGANG

LOCK=lockcode

Lockcode, der dem Transaktionscode eines Services als logisches Zahlenschloss zugeordnet ist. *lockcode* ist eine Zahl zwischen 1 und dem in der Anwendung erlaubten Maximalwert (MAX ...,KEYVALUE=*number*). Darf nicht zusammen mit dem Operanden ACCESS-LIST= angegeben werden.

Zur Zugriffskontrolle können Keysets sowohl für (UTM-)Benutzerkennungen (USER) als auch für die LTERM-/(OSI-)LPAP-Partner definiert werden. Ein mit Lockcode gesicherter Service kann dann nur gestartet werden, wenn im Keyset Benutzerkennung **und** im Keyset des LTERM-/(OSI-)LPAP-Partners ein mit dem Lockcode übereinstimmender Keycode enthalten ist.

Services, deren TACs nicht durch einen Lockcode oder eine ACCESS-LIST gesichert sind, können unter jeder Benutzerkennung und von jedem LTERM-/(OSI-)LPAP-Partner aus ohne Einschränkung aufrufen. Ausführliche Informationen zum Lock-/Keycode- sowie zum Access-List-Konzept finden Sie im openUTM-Handbuch „Konzepte und Funktionen“.

VORSICHT!
Wenn die Benutzerkennung und der LTERM-/(OSI-)LPAP-Partner nicht auch den Keycode für ein von diesem TAC aufgerufenes Folgeprogramm haben, bricht openUTM den Vorgang mit Fehler ab.

Standard: 0 (der TAC ist nicht mit Lockcode gesichert)
Maximalwert: Wert von MAX ...,KEYVALUE=*number*

PGWT dürfen Sie nur angeben, wenn in Ihrer Anwendung die Aufträge an TAC-Klassen prioritätengesteuert bearbeitet werden, d.h. die KDCDEF-Generierung enthält die Anweisung TAC-PRIORITIES.

Mit PGWT legen Sie fest, ob in einem Teilprogrammlauf, der für diesen Transaktionscode gestartet wird, blockierende Aufrufe (z.B. PGWT) durchgeführt werden dürfen.

YES Blockierende Aufrufe dürfen durchgeführt werden.
Geben Sie PGWT=YES an, dann müssen Sie den Transaktionscode einer TAC-Klasse zuordnen, d.h. Sie müssen TACCLASS= setzen.

Folgende Fälle sind zu beachten:

– CPI-C-Teilprogramme:
Soll ein CPI-C-Teilprogramm Dialog-Conversations unterhalten, bei denen das Senderecht durch den Aufruf Set_Send_Type mit send_type=CM_SEND_AND_PREP_TO_RECEIVE oder durch Aufruf von Receive im Zustand Send auf den Conversation-Partner übertragen wird, dann muss der Transaktionscode dieses CPI-C-Teilprogramms mit PGWT=YES generiert werden.

– XATMI-Teilprogramme:
Sobald in einer XATMI-Anwendung sowohl Requests als auch Conversational Services enthalten sind, müssen mindestens zwei Tasks gestartet werden und der Transaktionscode des Service muss mit PGWT=YES generiert werden.

NO Blockierende Aufrufe dürfen nicht durchgeführt werden.

Standard: NO

PROGRAM=objectname
Name des Teilprogramms, dem dieser TAC zugeordnet werden soll.

Für Asynchron- und Dialog-TACs muss ein Programmname generiert werden; für TAC-Queues ist der Parameter PROGRAM verboten.

Standard: Leerzeichen, d.h. kein Programmname

Falls das Programm im Betrieb der Anwendung nicht geladen ist oder die Zugriffsrechte den Aufruf nicht erlauben, dann ruft openUTM den Dialog-Vorgang BADTACS auf. Ist BADTACS in der Anwendung nicht generiert, dann gibt openUTM stattdessen die Meldung K009 aus.

QLEV=queue_level_number
(**q**ueue **lev**el)
gibt für Asychron-Transaktionscodes (TYPE=A) oder TAC-Queues (TYPE=Q) die maximale Anzahl der asynchronen Nachrichten an, die in der Message Queue dieses Transaktionscodes oder dieser TAC-Queue stehen können. Mit QLEV kann eine zu starke Belastung des Pagepool durch Aufträge für diesen TAC bzw. diese TAC-Queue verhindert werden.
openUTM berücksichtigt die Asynchron-Aufträge erst am Ende der Transaktion. Daher kann die in QLEV festgelegte Anzahl von Nachrichten für eine Message Queue überschritten werden, wenn in einer Transaktion mehrere Nachrichten für dieselbe Queue erzeugt wurden.

Soll nach dem Erreichen des QLEV eine weitere Nachricht erzeugt werden, so ist das Verhalten von openUTM abhängig von der Einstellung bei QMODE= (siehe dort).

Standard: 32767
Minimalwert: 0
Maximalwert: 32767 (bedeutet unbeschränkt)
Bei Verletzung des Maximalwertes wird von KDCDEF ohne Meldung der Standardwert gesetzt.

QMODE = (**Q**ueue **Mode**)
Bestimmt das Verhalten von openUTM für den Fall, dass bereits die maximal erlaubte Anzahl von Nachrichten in einer Queue gespeichert und somit der Queue-Level erreicht ist.

STD Ist zum Zeitpunkt des FPUT- oder DPUT-Aufrufs die Anzahl der in dieser Queue gespeicherten Nachrichten größer oder gleich der in QLEV= generierten Maximalzahl, so wird der FPUT- oder DPUT-Aufruf mit 40Z abgewiesen bzw. mit einer Meldung abgelehnt, wenn dieser TAC von einer Datensichtstation aus eingegeben wird.

WRAP-AROUND

nur für TACs mit TYPE=Q (TAC-Queues):
openUTM nimmt auch dann noch Nachrichten für diese Queue an, wenn der Queue Level bereits erreicht ist. Beim Schreiben dieser Nachricht in die Queue löscht openUTM die älteste der bereits in der Queue stehenden Nachrichten, deren Startzeitpunkt schon erreicht ist und die nicht gerade gelesen wird.

Standard: STD

Q-READ-ACL=read-keysetname

Dieser Parameter wird nur für TACs mit TYPE=Q (TAC-Queues) ausgewertet. Mit diesem Parameter werden die Rechte festgelegt, die ein Benutzer benötigt, um Nachrichten aus dieser Queue lesen und löschen zu können.

Bei diesem Parameter kann der Name eines KSETs angegeben werden, der mit einer KSET-Anweisung definiert ist. In diesem Fall kann ein Benutzer nur dann lesend auf diese TAC-Queue zugreifen, wenn der Schlüsselbund (KSET) des Benutzers und der Schlüsselbund des logischen Terminals, über das der Benutzer angemeldet ist, jeweils mindestens einen Schlüssel enthalten, der auch in dem hier angegebenen Schlüsselbund enthalten ist.

Wird bei Q-READ-ACL kein Schlüsselbund angegeben, dann können alle Benutzer Nachrichten aus dieser Queue lesen und dabei löschen.

Standard: kein Schlüsselbund

Q-WRITE-ACL=write-keysetname

Dieser Parameter wird nur für TACs mit TYPE=Q (TAC-Queues) ausgewertet. Er darf für die Dead Letter Queue nicht angegeben werden.
Mit diesem Parameter werden die Rechte festgelegt, die ein Benutzer benötigt, um Nachrichten in diese Queue schreiben zu können.

Bei diesem Parameter kann der Name eines KSETs angegeben werden, der mit einer KSET-Anweisung definiert ist. In diesem Fall kann ein Benutzer nur dann schreibend auf diese TAC-Queue zugreifen, wenn der Schlüsselbund (KSET) des Benutzers und der Schlüsselbund des logischen Terminals, über das der Benutzer angemeldet ist, jeweils mindestens einen Schlüssel enthalten, der auch in dem hier angegebenen Schlüsselbund enthalten ist.

Wird bei Q-WRITE-ACL kein Schüsselbund angegeben, dann können alle Benutzer Nachrichten in diese Queue schreiben.

Standard: kein Schlüsselbund

X/W X/W X/W X/W X/W X/W X/W X/W

RTIME=rtime Realzeit in Sekunden, die ein Teilprogramm maximal verbrauchen darf, wenn es über diesen TAC gestartet wird. Läuft das Teilprogramm länger, bricht openUTM den Vorgang ab und gibt eine Fehlermeldung aus (siehe openUTM-Handbuch „Meldungen, Test und Diagnose in Unix- und Windows-Systemen“).

rtime = 0 bedeutet, dass der Realzeit-Verbrauch des Teilprogramms nicht überwacht wird.

Standard: 0
Minimalwert: 0
Maximalwert: 32767

B B B

B B SATADM= legt fest, ob zum Aufruf des TACs die UTM-SAT-Administrationsberechtigung nötig ist.

B B B B YES Der TAC kann nur von Benutzern/Clients bzw. von Partner-Anwendungen aufgerufen werden, für die in der USER-, LPAP bzw. OSI-LPAP-Anweisung die Administrationsberechtigung für die SAT-Protokollierung (PERMIT=SATADM) generiert wurde.

B B NO Zur Nutzung des TACs muss der Benutzer/Client oder die Partner-Anwendung keine UTM-SAT-Administrationsberechtigung haben.

B B SATSEL= legt die Art der SAT-Protokollierung der Ereignisse beim Ablauf des Teilprogramms fest, das mit diesem TAC aufgerufen wird.

B B B Bei eingeschalteter SAT-Protokollierung (MAX ...,SAT=ON) werden dann während eines Programmlaufs unter diesem TAC die TAC-spezifischen Ereignisse entsprechend der Angaben für SATSEL protokolliert.

B B B B B B B B Die Art der SAT-Protokollierung kann allgemein für alle TACs und Benutzer in der SATSEL-Anweisung festgelegt werden. Der Operand SATSEL in der TAC-Anweisung legt zusätzlich zu den Angaben in der SATSEL-Anweisung eine TAC-spezifische Protokollierung fest. Wird die Protokollierung einer Ereignisklasse in der SATSEL-Anweisung ausgeschlossen, so werden Ereignisse dieser Klasse nicht protokolliert. (Zur Verknüpfung der EVENT-, TAC- und USER-spezifischen Einstellung der Protokollierung siehe openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“.)

B B B B B B B Die Anweisungen werden dann beim Start der Anwendung nicht wirksam. Sie erzeugen so eine Vorbelegung für eine SAT-Protokollierung, die im laufenden Betrieb bei Bedarf mit dem UTM-SAT-Administrationskommando KDCMSAT eingeschaltet werden kann.

B BOTH Es werden erfolgreiche **und** nicht erfolgreiche Ereignisse protokolliert.

B SUCC Es werden nur erfolgreiche Ereignisse protokolliert.

B FAIL Es werden nur nicht erfolgreiche Ereignisse protokolliert.

B NONE Es wird keine TAC-spezifische Art der SAT-Protokollierung definiert.

B Standard: NONE

STATUS= legt fest, ob der TAC oder die TAC-Queue beim Anwendungsstart gesperrt oder freigegeben ist.

ON Bedeutung für TACs: Der TAC ist nicht gesperrt und nach dem Start der Anwendung solange verfügbar, bis der Administrator ihn sperrt.

Bedeutung für TAC-Queues: Für diese Queue ist Schreiben und Lesen erlaubt.

Standard: ON

OFF Für TACs: Der TAC ist nach dem Start der Anwendung gesperrt. Aufträge für diesen TAC werden erst angenommen, wenn ihn der Administrator freigibt.

Handelt es sich um den TAC eines KDCS-Teilprogramms und ist er mit CALL=BOTH oder CALL=NEXT generiert, dann ist der TAC als Vorgangs-TAC (1. TAC eines Vorgangs) gesperrt, aber nicht als Folge-TAC eines Vorgangs.

Bedeutung für TAC-Queues: Die Queue ist für Schreibzugriffe gesperrt; Lesezugriffe sind erlaubt.

HALT Der TAC ist nach dem Start der Anwendung vollständig gesperrt, d.h. auch als Folge-TAC in einem Asynchron- oder Dialog-Vorgang.

Wenn dieser TAC als Folge-TAC aufgerufen wird, dann wird der Vorgang mit PEND ER (74Z) beendet. Der TAC muss vom Administrator freigegeben werden. Asynchron-Aufträge, die bereits in der Message Queue des TACs zwischengespeichert sind, werden nicht gestartet. Sie bleiben in der Message Queue, bis der Status des TACs vom Administrator wieder auf ON oder OFF gesetzt wird.

Bedeutung für TAC-Queues: Die Queue ist für Lese- und Schreibzugriffe gesperrt.

KEEP darf nur für TAC-Queues sowie für Asynchron-Transaktionscodes angegeben werden, die auch Vorgangs-TACs (CALL=BOTH oder CALL=FIRST) sind.
openUTM nimmt Aufträge für den Transaktionscode an. Die Aufträge werden jedoch nicht bearbeitet, sondern lediglich in die Message Queue des Transaktionscodes geschrieben. Sie werden bearbeitet, sobald der Administrator den Status des Transaktionscodes ändert in ON oder OFF.

STATUS=KEEP können Sie benutzen, um Aufträge zu sammeln, die erst zu einem Zeitpunkt ausgeführt werden sollen, an dem die Anwendung weniger belastet ist (z.B. nachts).

Um eine Überlastung des Pagepools durch zuviele zwischengespeicherte Aufträge zu vermeiden, sollten Sie die Auftragswarteschlange des Transaktionscodes durch den Parameter QLEV begrenzen.

Bedeutung für TAC-Queues: Die Queue ist gegen Lesen gesperrt; Schreiben ist erlaubt.

Für die Administrationskommandos KDCSHUT und KDCTAC wird der Status immer auf ON gesetzt, auch wenn Sie einen anderen Wert für STATUS angeben. Auf diese Weise bleibt Ihre Anwendung administrierbar.

TACCLASS= ordnet den Transaktionscode einer TAC-Klasse zu.

Die TAC-Klassen werden für die Steuerung der Bearbeitung von Dialog- und Asynchron-Aufträgen benötigt. Aufträge, die unterschiedlichen TAC-Klassen zugeordnet sind, werden von openUTM nach unterschiedlichen Kriterien gestartet. Über die TAC-Klasse, der ein Transaktionscode zugeordnet ist, wird gesteuert, ob ein Auftrag sofort bearbeitet oder zunächst in der Message Queue des Transaktionscodes zwischengespeichert wird, und wann er aus der Message Queue gelesen und bearbeitet wird. Für die Steuerung der Auftragsbearbeitung stehen zwei verschiedene Verfahren zur Verfügung (siehe Abschnitt „Auftragsbearbeitung über Prioritäten steuern“ auf Seite 208).

tacclass Die folgenden Zahlenwerte sind zulässig:

– 1 - 8 für Dialog-TACs
– 9 - 16 für Asynchron-TACs

Werden Asynchron-TAC-Klassen generiert, so muss der Wert MAX ...,ASYNTASKS einen Wert größer 0 haben.

Ist Ihre Anwendung **ohne** TAC-PRIORITIES-Anweisung generiert und durchläuft das zu diesem TAC gehörende Teilprogramm blockierende Aufrufe (z.B. den KDCS-Aufruf PGWT), dann müssen Sie für *tacclass* die Dialog- bzw. Asynchron-TAC-Klasse angeben, für die **TACCLASS PGWT=YES** gesetzt ist.

Ist Ihre Anwendung **mit** TAC-PRIORITIES-Anweisung generiert, dann können Sie diese TACs jeder beliebigen Dialog- bzw. Asynchron-TAC-Klasse zuordnen. Sie müssen dann lediglich **TAC ...,PGWT=YES** setzen.

Standard für Dialog-TACs:
Für Dialog-TACs erfolgt standardmäßig keine Zuordnung zu einer TAC-Klasse. Das zugehörige Teilprogramm wird gestartet, sobald ein Prozess die zugehörige Nachricht aus der Auftrags-Börse der Anwendung abholt.

Standard für Asynchron-TACs:
Für Asynchron-TACs wird als Standard der Wert 16 gesetzt.

Ist der Transaktionscode mit PGWT=YES generiert, dann müssen Sie den Transaktionscode einer TAC-Klasse zuordnen.

TACUNIT=tacunit

legt die Anzahl der Verrechnungseinheiten fest, die in der Abrechnungsphase des UTM-Accounting für jeden Aufruf dieses Transaktionscodes berechnet wird. Die Verrechnungseinheiten werden auf den Verrechnungseinheitenzähler der Benutzerkennung aufaddiert, die den Transaktionscode aufgerufen hat.
Dieser Operand ist nur notwendig, wenn openUTM Abrechnungsdaten sammeln soll (siehe auch ACCOUNT-Anweisung auf Seite 281 bzw. „Accounting“ im openUTM-Handbuch „Einsatz von openUTM-Anwendungen“).
Die Angabe ist nur ganzzahlig möglich.

Standardwert: 1
Minimalwert: 0
Maximalwert: 4095

B TCBENTRY=name_of_tcbentry_statement

B Nur relevant bei Transaktionscodes von Teilprogrammen, die mit PROGRAM ...,COMP=COB1 generiert sind.

B *name_of_tcbentry_statement* bezeichnet den Namen einer TCBENTRY-Anweisung, in der die TCB-Entries zusammengefasst wurden, die diesem TAC zugeordnet werden.

B Standard: Kein Name

B TIME= CPU-Resourcenverbrauch für Teilprogramm kontrollieren.

B time1 CPU-Zeit in Millisekunden, die das Teilprogramm mit diesem TAC während einer Verarbeitung maximal verbrauchen darf. Läuft das Teilprogramm länger, bricht openUTM den Vorgang mit einer Meldung ab: K017 bei Dialog-Programmen, K055 bei Asynchron-Programmen, KCRCCC ist 70Z, KCRCDC ist XT20 (siehe openUTM-Handbuch „Meldungen, Test und Diagnose in BS2000/OSD“).

B Der Wert 0 bedeutet, dass für das Teilprogramm, das über diesen TAC gestartet wurde, keine Zeitüberwachung erfolgen soll. Die Werte 1 bis 999 sind unzulässig und werden durch den Wert 1000 ersetzt.

B Standard: 30000 ms
B Minimalwert: 0 ms
B Maximalwert: 86400000 ms

VORSICHT!
Bei den Administrations-TACs KDCSHUT, KDCSHUTA, KDCDIAG und KDCDIAGA sollte der Wert von TIME=*time1* größer als der Standardwert gewählt werden (mindestens doppelt so groß; ≥ 60000 ms).

B Bei KDCSHUT WARN kann in Anwendungen mit vielen generierten Terminals mehr CPU-Zeit benötigt werden, als der Standardwert zulässt. Siehe dazu openUTM-Handbuch „Anwendungen administrieren“.
Analoges gilt, wenn Sie bei großen Anwendungen mit KDCDIAG DUMP=YES einen Diagnose-Dump anfordern.

B time2 Realzeit in Sekunden, die das Teilprogramm mit diesem TAC während einer Verarbeitung maximal verbrauchen darf. Läuft das Teilprogramm länger, bricht openUTM den Vorgang mit einer Meldung ab: K017 bei Dialog-Programmen, K055 bei Asynchron-Programmen, KCRCCC ist 70Z, KCRCDC ist XTA0 (siehe openUTM-Handbuch „Meldungen, Test und Diagnose in BS2000/OSD“). Der Wert 0 bedeutet, dass für das Teilprogramm, das über diesen TAC gestartet wurde, keine Überwachung des realen Zeitverbrauchs erfolgen soll.

B Standard: 0 s
Minimalwert: 0 s
Maximalwert: 32767 s

TYPE= gibt an, ob Aufträge an diesen Transaktionscode im Dialog oder asynchron bearbeitet werden oder ob eine TAC-Queue generiert wird.

D Dieser TAC ist ein Dialog-Transaktionscode, d.h. ein Auftrag mit diesem TAC wird im Dialog mit dem Auftraggeber bearbeitet.

Standard: D

A Dieser TAC ist ein Asynchron-Transaktionscode, d.h. ein Auftrag mit diesem TAC erzeugt einen Asynchron-Auftrag in der Message Queue des Transaktionscodes. Die Bearbeitung erfolgt entkoppelt vom Auftraggeber.

Q Mit dieser TAC-Anweisung wird eine TAC-Queue generiert. In eine solche Queue kann mit einem FPUT- oder DPUT-Aufruf eine Nachricht geschrieben und aus der Queue kann mit einem DGET-Aufruf gelesen werden.

TACCLASS - Prozess-Anzahl für TAC-Klassen festlegen

Mit der Steueranweisung TACCLASS legen Sie fest, nach welchem Verfahren die Auftragsbearbeitung in dieser UTM-Anwendung gesteuert werden soll. Das heißt, Sie legen die Kriterien fest, nach denen Aufträge für Transaktionscodes, die einer TAC-Klasse zugeordnet sind, von openUTM gestartet werden sollen.

Die Festlegung dieser Kriterien kann alternativ mit der TACCLASS-Anweisung oder der TAC-PRIORITIES-Anweisung erfolgen.

Eine TAC-Klasse besteht dabei aus einer Teilmenge der generierten Transaktionscodes der Anwendung. Die Aufteilung der TACs in TAC-Klassen wird in der TAC-Anweisung mit dem Operanden TACCLASS= vorgenommen.

Indem Sie mindestens eine TACCLASS-Anweisung generieren, legen Sie fest, dass die Auftragsbearbeitung in Ihrer Anwendung durch die Beschränkung der Prozesszahl für die einzelnen TAC-Klassen gesteuert wird. Sie dürfen dann keine TAC-PRIORITIES-Anweisung absetzen.

In der TACCLASS-Anweisung legen Sie fest, wieviele Prozesse der UTM-Anwendung gleichzeitig für die TACs einer TAC-Klasse arbeiten dürfen. Darüber hinaus können Sie im Operanden PGWT festlegen, ob in Teilprogrammläufen, die von Transaktionscodes der TAC-Klasse gestartet werden, blockierende Aufrufe (z.B. der KDCS-Aufruf PGWT) zulässig sind oder nicht. Sie dürfen nur einer Dialog- und einer Asynchron-TAC-Klasse die Eigenschaft PGWT=YES, d.h. blockierende Aufrufe sind erlaubt, zuordnen.

Die Zahl der Prozesse einer TAC-Klasse, die Sie in der TACCLASS-Anweisung festlegen, kann der Administrator verändern (siehe openUTM-Handbuch „Anwendungen administrieren“).

Mit der TACCLASS-Anweisung können Sie beeinflussen, wie die UTM-Anwendung durch Teilprogramme einzelner TACs belastet werden darf. Sie können z.B. verhindern, dass langlaufende Teilprogramme die Anwendung blockieren**.** Werden Asynchron-Vorgänge bei der verteilten Verarbeitung verwendet, dann können Sie verhindern, dass alle Prozesse der Anwendung, die für Asynchron-Verarbeitung zur Verfügung stehen, gleichzeitig durch diese Vorgänge belegt werden.

Standardwerte

Alle TAC-Klassen werden bei der KDCDEF-Generierung implizit erzeugt, wenn Sie einen Transaktionscode mit der Anweisung TAC ...,TACCLASS= oder mit TACCLASS eine TAC-Klasse generieren.

Setzen Sie **keine** TAC-PRIORITIES-Anweisung ab, dann sollten Sie für jede verwendete TAC-Klasse eine TACCLASS-Anweisung schreiben. In diesem Fall ordnet openUTM den TAC-Klassen, für die keine TACCLASS-Anweisung abgesetzt wird, die Minimalwerte für TASKS und TASKS-FREE zu. Für TAC-Klassen mit PGWT=YES müssen Sie dann immer TACCLASS-Anweisungen absetzen!

Wenn Sie keine TAC-Klassen verwenden, d.h. es ist in keiner TAC-Anweisung der Operand TACCLASS= angegeben und es gibt weder eine TACCLASS- noch TAC-PRIORITIES-Anweisung, dann gilt:

- Dialog-TACs werden ohne Einschränkung verarbeitet.
- Für Asynchron-TACs gilt als Einschränkung die Prozess-Anzahl, die im Startparameter ASYNTASKS angegeben wird. Dieser Wert kann durch die Administration geändert werden.

Ein ausführliche Beschreibung über TAC-Klassen und Prioritätensteuerung finden Sie im Abschnitt „Auftragssteuerung - Prioritäten und Prozessbeschränkung“ auf Seite 205.

```
TACCLASS␣     tacclass
                ,{ TASKS=number1 | TASKS-FREE=number2 }
              [ ,PGWT={ NO | YES } ]
```

tacclass

Nummer der TAC-Klasse, für die die Prozess-Anzahl festgelegt werden soll. Folgende TAC-Klassen können angegeben werden:

– die Dialog-TAC-Klassen 1 - 8.
– die Asynchron-TAC-Klassen 9 - 16.

Sie ordnen einen Transaktionscode dieser TAC-Klasse zu, indem Sie in der entsprechenden TAC-Anweisung TACCLASS=*tacclass* angeben.

Eine Asynchron-TAC-Klasse dürfen Sie nur angeben, wenn Sie in MAX ...,ASYNTASKS einen Wert ungleich 0 generieren.

Asynchron-Transaktionscodes, die keiner TAC-Klasse zugeordnet sind, werden automatisch in die TAC-Klasse 16 eingeordnet.

Die Nummer der TAC-Klasse ist keine Priorität, sondern nur eine Bezeichnung für die TAC-Klasse.
Es hängt nur von der Zahl *number1* der erlaubten Prozesse ab, wie stark die Verarbeitung einer TAC-Klasse gedrosselt wird und somit TACs anderer Klassen schneller bearbeitet werden. Diese Drosselung kann für eine TAC-Klasse nur wirken, wenn die Zahl *number1* kleiner ist als die Anzahl der laufenden Prozesse der Anwendung.

TASKS=number1

gibt an, wieviele Prozesse der Anwendung höchstens gleichzeitig für die TACs dieser Klasse arbeiten dürfen. Die erlaubten Werte für TASKS= sind abhängig vom Wert des Operanden PGWT und der Operanden TASKS, TASKS-IN-PGWT und ASYNTASKS der MAX-Anweisung. Die erlaubten Wertebereiche für TASKS=*number1* entnehmen Sie bitte der Tabelle:

	Klasse 1 - 8 Dialog-TACs		**Klasse 9 - 16 Asynchron-TACs**	
Minimal-wert	PGWT =NO	PGWT=YES	PGWT= NO	PGWT=YES
	1	1	0	0
Maxi-malwert	TASKS *)	TASKS-IN-PGWT *)	ASYNTASKS*)	der kleinere der Werte: ASYNTASKS, *) TASKS-IN-PGWT*)

*) wie in der MAX-Anweisung angegeben

Pflichtoperand, wenn TASKS-FREE= nicht angegeben wird.

Wenn Sie für eine Dialog-TAC-Klasse TASKS=0 angeben, ersetzt openUTM diesen Wert automatisch durch 1.

i Die Summe der Angaben bei TASKS= in den einzelnen TACCLASS-Anweisungen darf größer sein als die maximal zulässige Anzahl von Prozessen der Anwendung (MAX ...,TASKS=).

TASKS-FREE=number2

Im Operanden TASK-FREE wird festgelegt für

- Dialog-TAC-Klassen:
  wieviele Prozesse der UTM-Anwendung mindestens für die Verarbeitung von TACs anderer Klassen freigehalten werden sollen,
- Asynchron-TAC-Klassen:
  wieviele der Prozesse, die für Asynchron-Aufträge zugelassen sind (MAX ...,ASYNTASKS=), mindestens für die Verarbeitung von TACs anderer Asynchron-TAC-Klassen freigehalten werden sollen.

Gegenüber der Verwendung des Parameters TASKS bietet TASKS-FREE den Vorteil, dass bei einer Änderung der gesamten Prozessanzahl der Anwendung im laufenden Betrieb die Anzahl der für eine TAC-Klasse erlaubten Prozesse dynamisch angepasst wird.

Die erlaubten Werte für TASKS-FREE=*number2* sind abhängig vom Wert der Operanden TASKS und ASYNTASKS der MAX-Anweisung. Die erlaubten Wertebereiche für TASKS-FREE= entnehmen Sie bitte der Tabelle:

	Klasse 1 - 8 Dialog-TACs	**Klasse 9 - 16 Asynchron-TACs**
Minimalwert	1	1
Maximalwert	TASKS-1 *)	ASYNTASKS *)

*) wie in der MAX-Anweisung angegeben

Pflichtoperand, wenn TASKS= nicht angegeben wird.

Wenn Sie TASKS-FREE=0 angeben, ersetzt openUTM diesen Wert automatisch durch 1.

PGWT= (**p**ro**g**ram **w**ai**t**)
gibt an, ob in dieser TAC-Klasse Teilprogramme ablaufen dürfen, die blockierende Aufrufe enthalten, z.B. KDCS-Aufruf PGWT.
(PGWT, siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“ und openUTM-Handbuch „Konzepte und Funktionen“).

YES In dieser TAC-Klasse sind blockierende Aufrufe zulässig.
PGWT=YES darf nur generiert werden, wenn MAX ...,TASKS-IN-PGWT≠0 definiert wurde. Die Angabe von PGWT=YES ist höchstens für eine Dialog-TAC-Klasse und eine Asynchron-TAC-Klasse zulässig. Teilprogramme, die PGWT-Aufrufe enthalten, müssen diesen TAC-Klassen zugeordnet werden.

– CPI-C-Teilprogramme:
Soll ein CPI-C-Teilprogramm Dialog-Conversations unterhalten, bei denen das Senderecht durch den Aufruf Set_Send_Type mit send_type=CM_SEND_AND_PREP_TO_RECEIVE oder durch Aufruf von Receive im Zustand Send auf den Conversation-Partner übertragen wird, so muss der Transaktionscode dieses CPI-C-Teilprogramms einer TAC-Klasse zugeordnet werden, die mit PGWT=YES generiert ist, beispielsweise:

```
MAX TASKS=2
MAX TASKS-IN-PGWT=1
TACCLASS 1,TASKS=1,PGWT=YES
TAC CPIC1,PROGRAM=xyz,API=(XOPEN,CPIC),TACCLASS=1
```

– XATMI-Teilprogramme:
Sobald in einer XATMI-Anwendung sowohl Requests als auch Conversational Services enthalten sind, müssen mindestens zwei Tasks gestartet werden und es muss eine TAC-Klasse generiert werden, für die PGWT-Aufrufe erlaubt sind. Ein Service ist immer an die Task gebunden. Bei einer Anwendung, die ausschließlich Request/Response-Services enthält, ist dies nicht notwendig.

NO Teilprogramme, die blockierende Aufrufe enthalten, sind in dieser TAC-Klasse nicht zulässig.

Standard: NO

Erst wenn in den Message Queues der Dialog-TAC-Klassen keine Aufträge mehr zwischengespeichert sind, werden blockierende Aufrufe für Asynchron-TACs abgearbeitet.

Beispiel

In der folgenden Tabelle wird dargestellt, wie sich die Angabe von TASKS-FREE unter entsprechenden Rahmenbedingungen auf die einer TAC-Klasse zur Verfügung stehenden Prozesszahlen auswirkt.

- In der Spalte CURRENT TASKS ist die jeweils aktuelle maximale Zahl der Prozesse aufgeführt, die der UTM-Anwendung zur Verfügung stehen. CURRENT ASYNTASKS enthält die jeweils aktuelle Obergrenze für Prozesse zur Bearbeitung von asynchronen Vorgängen. Die anwendungsglobalen Maximalwerte für CURRENT TASKS bzw. CURRENT ASYNTASKS werden mit den Operanden TASKS bzw. ASYNTASKS der MAX-Anweisung festgelegt. Während des Anwendungslaufs können die aktuellen Werte im Rahmen dieser Obergrenze dynamisch mit den Operanden TASKS bzw. MAXASYN des KDCAPPL-Kommandos geändert werden.
- Die Spalte DIALOG enthält die Anzahl von Prozessen, die maximal für eine bestimmte DIALOG-TAC-Klasse zur Verfügung steht, wenn für diese TAC-Klasse TASKS-FREE=*nn* angegeben wurde.

- Die Spalte ASYNCH enthält die Anzahl von Prozessen, die maximal für eine bestimmte Asynchron-TAC-Klasse zur Verfügung steht, wenn für diese TAC-Klasse TASKS-FREE=*number2* angegeben wurde.

CURRENT TASKS	CURRENT ASYNTASKS	TASKS-FREE	DIALOG	ASYNCH
10	9	2	8	7
6	6	2	4	4
3	3	2	1	1
2	2	2	1	0
1	1	2	1	0
10	5	3	7	2
6	5	3	3	2

TAC-PRIORITIES - Prioritäten der TAC-Klassen festlegen

Mit der Steueranweisung TAC-PRIORITIES legen Sie fest, nach welchem Verfahren die Auftragsbearbeitung in dieser UTM-Anwendung gesteuert werden soll. Das heißt, Sie legen die Kriterien fest, nach denen Aufträge für Transaktionscodes, die einer TAC-Klasse zugeordnet sind, von openUTM gestartet werden sollen.

Die Festlegung dieser Kriterien kann alternativ mit der TAC-PRIORITIES-Anweisung oder der TACCLASS-Anweisung erfolgen.

Eine TAC-Klasse besteht dabei aus einer Teilmenge der generierten Transaktionscodes der Anwendung. Die Aufteilung der Transaktionscodes in TAC-Klassen wird in der TAC-Anweisung mit dem Operanden TACCLASS= vorgenommen.
Wird der TACCLASS-Operand nicht angegeben, so werden Dialog-TACs keiner TAC-Klasse und Asynchron-TACs der Asynchron-TAC-Klasse 16 zugewiesen.

Mit TAC-PRIORITIES legen Sie im Einzelnen fest:

- dass die Verteilung der Prozesse auf die TAC-Klassen prioritätengesteuert erfolgen soll. Sie dürfen dann keine TACCLASS-Anweisung absetzen.
- nach welchen Algorithmen die zur Verfügung stehenden Prozesse der Anwendung auf die Dialog- bzw. Asynchron-TAC-Klassen verteilt werden sollen.
  Operanden: DIAL-PRIO und ASYN-PRIO
- wieviele Prozesse der Anwendung maximal Aufträge an Dialog-TAC-Klassen bearbeiten dürfen.
  Operand: FREE-DIAL-TASKS

Prioritäten für die TAC-Klassen festlegen

Bei der Prioritätensteuerung können Sie für Dialog- und für Asynchronaufträge jeweils zwischen absoluten, relativen oder gleichen Prioritäten wählen. Die Steuerung der Auftragsbearbeitung von Dialog- und Asynchron-Aufträgen erfolgt unabhängig voneinander.

Aufträge für Dialog-TACs, die keiner TAC-Klasse zugeordnet sind, werden unabhängig von den für Dialog-Aufträge eingestellten Prioritäten bearbeitet. Diese Aufträge werden immer sofort nach ihrer Entgegennahme aus dem Transportsystem gestartet.

Bei absoluten und relativen Prioritäten spielt die Nummer der TAC-Klasse eine Rolle. Aufträge an TAC-Klassen mit einer niedrigeren Nummer haben eine höhere Priorität als Aufträge an TAC-Klassen mit einer höheren Nummer. Bei den Dialog-TAC-Klassen hat also die TAC-Klasse 1 die höchste und die TAC-Klasse 8 die niedrigste Priorität. Bei den Asynchron-TAC-Klassen hat die TAC-Klasse 9 die höchste und die TAC-Klasse 16 die niedrigste Priorität.

Bei der Verwendung von absoluten Prioritäten werden freie und für die TAC-Klassen-Verarbeitung zur Verfügung stehende Prozesse der Anwendung immer der TAC-Klasse mit der höchsten Priorität, also 1 bzw. 9 zugeordnet, sofern es wartende Aufträge für diese TAC-Klasse gibt.
Erst wenn es keine wartenden Aufträge mehr in der TAC-Klasse mit der höchsten Priorität gibt, werden wartende Aufträge der TAC-Klasse mit der nächst niedrigeren Priorität bearbeitet.

Wollen Sie verhindern, dass wartende Aufträge einer TAC-Klasse mit niedriger Priorität eventuell lange Zeit nicht bearbeitet werden, so sollten Sie relative Prioritäten verwenden. Bei relativen Prioritäten werden Aufträge aus TAC-Klassen hoher Priorität häufiger bearbeitet als Aufträge aus TAC-Klassen niedrigerer Priorität.

Bei gleichen Prioritäten werden, sofern wartende Aufträge vorhanden, aus jeder TAC-Klasse gleich viele Aufträge bearbeitet.

Anzahl der Prozesse beschränken, die Aufträge an TAC-Klassen bearbeiten

Auch bei der Prioritätensteuerung der TAC-Klassen können Sie die Anzahl der Prozesse, die Aufträge der TAC-Klassen bearbeiten, beschränken, um Prozesse für administrative Aufgaben oder interne Aufträge frei zu halten.

Mit dem Operanden FREE-DIAL-TASKS beschränken Sie die Prozesszahl für die Dialog-TAC-Klassen relativ zur Gesamtanzahl der Prozesse.
Mit MAX ASYNTASKS=(*atask_number*,...) beschränken Sie die Prozesszahl für Asynchron-TAC-Klassen.

Diese Beschränkung ist aber jeweils gleich für alle Asynchron- bzw. für alle Dialog-TAC-Klassen.

Transaktionscodes, die Teilprogrammläufe mit blockierenden Aufrufen starten

Bei Verwendung der Anweisung TAC-PRIORITIES dürfen Transaktionscodes mit blockierenden Aufrufen (z.B. KDCS-Aufruf PGWT) jeder TAC-Klasse zugeordnet werden, sofern der Operand TASKS-IN-PGWT der MAX-Anweisung mit einem Wert > 0 generiert wird. Für diese Transaktionscodes müssen Sie TAC PGWT=YES generieren.

Wird bei der Generierung weder eine TACCLASS-Anweisung noch die Anweisung TAC-PRIORITIES abgesetzt, obwohl bei mindestens einer TAC-Anweisung der TACCLASS-Parameter angegeben wurde, dann wirken die Standardwerte der TACCLASS-Anweisung. TAC-Prioritäten werden in diesem Fall nicht angewendet. Siehe dazu die Beschreibung von TACCLASS auf Seite 506.

Eine ausführliche Beschreibung über TAC-Klassen und Prioritätensteuerung finden Sie im Abschnitt „Auftragssteuerung - Prioritäten und Prozessbeschränkung“ auf Seite 205.

```
TAC-PRIORITIES␣ [DIAL-PRIO={ABS | REL | EQ } ]
                [ ,ASYN-PRIO = { ABS | REL | EQ } ]
                [ ,FREE-DIAL-TASKS = number ]
```

DIAL-PRIO = gibt an, nach welcher Priorität freie Prozesse auf die Dialog-TAC-Klassen mit wartenden Aufträgen verteilt werden sollen. Zu wartenden Dialog-Aufträgen kommt es nur dann, wenn zu einer Zeit mehr Aufträge an der Auftragsbörse abgeholt werden, als Prozesse für die Dialog-TAC-Klassen zur Verfügung stehen. Die Aufträge werden dann in die Auftrags-Queues der Transaktionscodes geschrieben, aus der sie von den freiwerdenden Prozessen entsprechend ihrer Priorität gelesen und bearbeitet werden.

ABS Absolute Priorität:
Ein freier Prozess wird immer der TAC-Klasse mit der höchsten Priorität (TAC-Klasse 1) zugeordnet, sofern es für diese TAC-Klasse wartende Aufträge gibt. TAC-Klassen mit niedrigerer Priorität werden nur bedient, wenn es in allen TAC-Klassen höherer Priorität keine wartenden Aufträge mehr gibt.

REL Relative Priorität:
Freie Prozesse werden öfter TAC-Klassen hoher als TAC-Klassen niedriger Priorität zugeordnet, sofern für diese wartende Aufträge vorhanden sind. Sind für alle Dialog-TAC-Klassen Aufträge vorhanden, so wird ein freiwerdender Prozess der TAC-Klasse 1 doppelt so oft wie der TAC-Klasse 2 zugeordnet und der TAC-Klasse 2 werden Prozesse wiederum doppelt so häufig zugeordnet wie der TAC-Klasse 3 usw.

EQ Gleiche Priorität:
Sofern Aufträge vorhanden sind, werden alle TAC-Klassen gleich häufig bedient. Diese Gleichverteilung kann gestört werden, wenn eine TAC-Klasse zeitweise keine wartenden Aufträge enthält oder in ihr häufiger Teilprogrammläufe mit blockierenden Aufrufen (z.B. KDCS-Aufruf PGWT) auftreten.

Standard: EQ

ASYN-PRIO= gibt an, nach welchen Prioritäten Prozesse auf die Asynchron-TAC-Klassen mit ablaufbereiten Asynchron-Aufträgen oder unterbrochenen Asynchron-Aufträgen verteilt werden sollen.

Ist die maximale Anzahl gleichzeitig offener Asynchron-Vorgänge erreicht (gesetzt in MAX ASYNTASKS=(...,*service_number*)), so wird kein neuer anstehender Asynchron-Auftrag mehr gestartet, sondern ein unterbrochener offener Asynchron-Vorgang nach Priorität ausgewählt und fortgesetzt.

ABS Absolute Prioritäten:
Ein freier Prozess wird immer der TAC-Klasse mit der höchsten Priorität, also neun zugeordnet, sofern es dort anstehende Asynchron-Aufträge oder unterbrochene Asynchron-Aufträge gibt. Freiwerdende Prozesse bearbeiten Aufträge einer TAC-Klasse mit niedrigerer Priorität erst dann, wenn in den Message Queues aller TAC-Klassen höherer Priorität keine anstehenden oder unterbrochenen Asynchron-Aufträge mehr stehen.

REL Relative Prioritäten:
Freie Prozesse werden öfter TAC-Klassen hoher als TAC-Klassen niedriger Priorität zugeordnet, sofern für die TAC-Klassen höherer Priorität anstehende oder unterbrochene Aufträge vorhanden sind. Sind für alle TAC-Klassen Aufträge vorhanden, dann werden freiwerdende Prozesse der TAC-Klasse 9 doppelt so oft zugeordnet wie der TAC-Klasse 10, und der TAC-Klasse 10 wiederum doppelt so häufig wie der TAC-Klasse 11 usw.

EQ Gleiche Prioritäten:
Sofern Aufträge vorhanden sind, werden alle TAC-Klassen gleich häufig bedient. Diese Gleichverteilung kann in der Praxis gestört werden, wenn etwa eine TAC-Klasse zeitweise keine wartenden Aufträge enthält oder in ihr häufiger Teilprogrammläufe mit blockierenden Aufrufen (z.B. KDCS-Aufruf PGWT) auftreten.

Standard: EQ

FREE-DIAL-TASKS=number

Mit FREE-DIAL-TASKS beschränken Sie die Gesamtanzahl der Prozesse, die Aufträge an Dialog-TAC-Klassen bearbeiten dürfen, relativ zur Anzahl aller Prozesse der Anwendung. In *number* geben Sie die Anzahl der Prozesse der Anwendung an, die mindestens für die Verarbeitung von Aufträgen freigehalten werden sollen, die keiner Dialog-TAC-Klasse angehören.

Die maximale Anzahl der Prozesse, die gleichzeitig Asynchron-Aufträge bearbeiten dürfen, wird durch FREE-DIAL-TASKS= nicht beschränkt. Dazu steht der MAX-Operand ASYNTASKS=*atask_number* zur Verfügung.

Minimalwert: 0 (keine Einschränkung)
Maximalwert: TASKS - 1 (TASKS aus MAX-Anweisung)
Standardwert: 1

Beispiel
In der MAX-Anweisung wurde TASKS=7 und ASYNTASKS=2 gesetzt. In der Anweisung TAC-PRIORITIES wird FREE-DIAL-TASKS=3 generiert. Die Anwendung wird mit sechs Prozessen betrieben. Dann können maximal drei Prozesse Aufträge der TAC-Klassen 1 bis 8 und maximal zwei Prozesse Aufträge der TAC-Klassen 9 bis 16 bearbeiten. Ein Prozess bleibt für Dialog-Aufträge reserviert, die keiner TAC-Klasse zugeordnet sind.

TCBENTRY - Gruppe von TCB-Entries definieren (BS2000/OSD)

B Die TBCENTRY-Anweisung ist nur für COB1-Teilprogramme zulässig. COBOL-Teilprogramme, die nicht ILCS-fähig sind, müssen bei openUTM mit PROGRAM ...,COMP=COB1 generiert werden. TCB-Entries sind von Vorteil in Verbindung mit COBOL-DML sowie bei einem GOTO in eine PERFORM-Routine. Näheres hierzu ist dem openUTM-Handbuch „Anwendungen programmieren mit KDCS“ zu entnehmen.

B TCB-Entries dienen dazu, verschachtelte COBOL-Programme reentrant-fähig zu machen. In den folgenden Fällen sind TCB-Entries notwendig:

- B In Verbindung mit COBOL-DML:
  Wenn in einer DECLARATIVES-Unterabteilung die Klausel USE-DATABASE-EXEPTION verwendet und der Programmablauf innerhalb dieser Declaratives mit PEND beendet wird. In diesem Fall muss der TCB-Entry I$ITCUPS angegeben werden, weil sonst der Zähler für den Durchlauf der DECLARATIVES nicht zurückgesetzt wird, was zu einer COBOL-Fehleraktion führt. I$ITCUPS setzt daher den Zähler bei einem PEND innerhalb der DECLARATIVES zurück.

- B Bei einem GOTO in eine PERFORM-Routine:
  Wenn ein Teilprogramm in einer PERFORM-Routine beendet wird, merkt sich das COBOL-Laufzeitsystem die Rücksprungadresse. Springt man im nächsten Teilprogramm mit GOTO in die PERFORM-Routine, dann verhält sich das Teilprogramm so, als wäre die PERFORM-Routine noch offen und springt auf die Rücksprungadresse. Durch Angabe eines TCB-Entries (mit beliebigem Namen) werden die Merker zurückgesetzt.

B TCB-Entries müssen auch dem COB1-Compiler über einen COBRUN-Parameter bekannt gemacht werden.

B Die Anweisung kann mehrmals angegeben werden.

```
B  TCBENTRY␣     tcbentry_groupname
B                 ,ENTRY=( entry1,..., entry18 )
```

B tcbentry_groupname
ist ein frei wählbarer, max. 8 Zeichen langer Name, über den die mit dieser TCBENTRY-Anweisung definierte Gruppe von TCB-Entries angesprochen werden kann. Mit diesem Namen kann eine Gruppe von TCB-Entries an eine TAC-Anweisung geknüpft werden.

B ENTRY= TCB-Entry-Name

TLS - Namen von TLS-Blöcken festlegen

Jedem LTERM-Partner kann ein Terminal-spezifischer Langzeitspeicher TLS zugeordnet werden, der aus mehreren Blöcken bestehen kann. Der Name eines TLS-Blockes wird mit der TLS-Anweisung festgelegt. Ein TLS-Block wird identifiziert über den Namen *ltermname* des LTERM-Partners und den Blocknamen. openUTM stellt damit jedem LTERM-Partner einen TLS-Block mit diesem Namen zur Verfügung. Durch Wiederholen der TLS-Anweisung mit unterschiedlichen Blocknamen können mehrere Blöcke pro LTERM-Partner definiert werden.

Die mit der TLS-Anweisung definierten TLS-Blöcke werden auch den LPAP-Partnern und OSI-LPAP-Partnern bei der verteilten Verarbeitung zugeordnet.

Es dürfen maximal 100 TLS-Anweisungen angegeben werden.

```
TLS␣          name
```

name max. 8 Zeichen langer Name eines TLS-Blocks.

TPOOL - LTERM-Pools definieren

Mit der TPOOL-Anweisung definieren Sie einen L**T**ERM-**Pool** und legen dessen Eigenschaften fest. Über einen LTERM-Pool können sich unterschiedliche Clients mit gleichen technischen Eigenschaften (Partner- und Prozessortyp) über LTERM-Partner an eine UTM-Anwendung anschließen. Drucker werden dabei **nicht** unterstützt. In der TPOOL-Anweisung wird nur der Typ (PTYPE=) und der Rechnername (PRONAM=) für den Client angegeben. Die Zuordnung LTERM-Partner zu Client erfolgt dynamisch beim Verbindungsaufbau durch den UTM-Systemcode, und zwar über den in der TPOOL-Anweisung definierten Namen des LTERM-Partners und den Namen des Clients. Die Zuordnung besteht nur für die Dauer einer Session, es besteht keine statische Zuordnung wie im Anweisungspaar LTERM / PTERM. Bei einem LTERM-Pool müssen die Clients nicht explizit in der Anwendung konfiguriert sein (keine Definition eines PTERM). Es können sich gleichzeitig soviele Clients anschließen, wie LTERM-Partner im LTERM-Pool generiert sind.

Bei Clients, die sich über einen LTERM-Pool anschließen (d.h. die nicht explizit generiert sind), kann der Verbindungsaufbau nur „von außen“ angestoßen werden, d.h. vom Client selbst. Der Verbindungsaufbau über UTM- Administrationskommandos ist also nicht möglich.

B B Unter BS2000/OSD ist auch der Verbindungsaufbau über BCAM-Administrationskommandos oder durch vordefinierte BCAM-Verbindungen nicht möglich.

Sie können mit der TPOOL-Anweisung LTERM-Pools mit verschiedenen Freiheitsgraden für den Verbindungsaufbau definieren:

- Mit PRONAM=*processorname* und PTYPE=*partnertyp* wird der LTERM-Pool so generiert, dass nur Clients gleichen Typs, die sich auf dem angegebenen Rechner befinden, über diesen LTERM-Pool Verbindungen zu der UTM-Anwendung aufbauen.
- Mit PRONAM=*ANY können sich alle Clients eines Typs an die UTM-Anwendung anschließen, unabhängig davon, auf welchem Rechner sie sich befinden.
- B B B B Mit PTYPE=*ANY definieren Sie unter BS2000/OSD einen LTERM-Pool ohne spezifischen Clienttyp.
  Über einen solchen LTERM-Pool können Clients beliebigen Typs, die sich auf dem angegebenen Rechner befinden, Verbindungen zu der UTM-Anwendung aufbauen.
- B B B Mit PRONAM=*ANY und PTYPE=*ANY können sich beliebige Clients auf beliebigen Rechnern an eine UTM-Anwendung unter BS2000/OSD anschließen (offener LTERM-Pool).

Sie können mehrere LTERM-Pools definieren, d.h. mehrere TPOOL-Anweisungen pro KDCDEF-Lauf angeben. Es ist jedoch Folgendes zu beachten:

B
- *BS2000/OSD:*

  Für LTERM-Pools, für die mit PROTOCOL=STATION das Benutzerprotokoll NEABT vereinbart wurde, muss die Kombination PRONAM / PTYPE / BCAMAPPL eindeutig sein. Bei LTERM-Pools mit PROTOCOL=NO muss die Kombination PRONAM / BCAMAPPL eindeutig sein.

  Der Client muss immer das Benutzerdienstprotokoll unterstützen, das in der TPOOL-Anweisung angegeben wurde. Für Clients mit PTYPE=APPLI, PTYPE=SOCKET oder PTYPE=UPIC-R wird stets PROTOCOL=NO generiert. Für LTERM-Pools, die mit PTYPE=*ANY generiert sind, muss immer PROTOCOL=STATION angegeben werden.

  Beim Verbindungsaufbau übernimmt openUTM den Typ (PTYPE) des Clients, der mit PTYPE=*ANY generiert ist, aus dem Benutzerdienstprotokoll (Connection Letter). openUTM überprüft, ob der Typ unterstützt werden kann. Unterstützt openUTM den Typ nicht, dann wird der Verbindungswunsch abgelehnt.

X/W
- *Unix-, Windows-Systeme:*

  Für die LTERM-Pools muss die Kombination PRONAM/PTYPE/BCAMAPPL eindeutig sein.

Die LTERM-Partner eines LTERM-Pools werden mit LTERM ...,RESTART=NO generiert. Beim Verbindungsaufbau werden daher alle asynchronen Nachrichten gelöscht, die in der Message Queue der LTERM-Partner des LTERM-Pools zwischengespeichert sind. In Anwendungen, die ohne Benutzerkennungen generiert sind, kann für Clients, die über einen LTERM-Pool mit der Anwendung verbunden sind, nach einem Verbindungsabbau und nachfolgendem Verbindungswiederaufbau kein Vorgangswiederanlauf durchgeführt werden.

Für einen LTERM-Pool können Sie Zugriffsrechte festlegen (Operand KSET), die die über den LTERM-Pool angeschlossenen Clients ausüben können. In Anwendungen mit Benutzerkennungen können Sie für LTERM-Pools, die zum Anschluss von UPIC-Clients oder TS-Anwendungen generiert werden, die mit KSET festgelegten Zugriffsrechte mit dem Operanden USER-KSET einschränken. Die Zugriffsrechte in KSET beziehen sich dann auf Clients, die bei der Anmeldung explizit eine Benutzerkennung angeben. Die eingeschränkten Zugriffsrechte in USER-KSET werden wirksam, wenn der Client bei der Anmeldung keine Benutzerkennung übergibt, d.h. die „Verbindungs-Benutzerkennung“ aktiv ist.

B B Für jeden LTERM-Pool können Sie mit dem Operanden LOCALE eine clientspezifische Sprachumgebung definieren.

```
TPOOL␣      [ ,BCAMAPPL=local_appliname ]
            [ ,CONNECT-MODE={ SINGLE | MULTI }
            [ ,ENCRYPTION-LEVEL={ NONE | 1 | 2 | 3 | 4 | TRUSTED } ]
            [ ,IDLETIME=time ]
            [ ,KSET=keysetname1 ]
            [ ,LOCK=lockcode ]
              ,LTERM=ltermprefix
            [ ,MAP={ USER | SYSTEM | SYS1[1] | SYS2[1] | SYS3[1] | SYS4[1]} ]
              ,NUMBER=number1
              ,PRONAM={ processorname | C'processorname' | *ANY }
                                  nur in BS2000/OSD Pflichtoperand
              ,PTYPE={ partnertyp | *ANY[2] }
            [ ,QLEV=queue_level_number ]
            [ ,STATUS=( { ON | OFF }, number2 ) ]
            [ ,TERMN=termn_id ]
            [ ,USER-KSET=keysetname2 ]
            [ ,USP-HDR={ALL | MSG | NO}]

B           BS2000/OSD-spezifische Operanden
B           [  ANNOAMSG={ Y | N } ]
B           [ ,FORMAT= { + | * | # }formatname ]
B           [ ,KERBEROS-DIALOG={ YES | NO } ]
B           [ ,LOCALE=( [ lang_id ][,[ terr_id ][ ,ccsname ] ] ) ]
B           [ ,NETPRIO={ MEDIUM | LOW }
B           [ ,PROTOCOL={ N | STATION } ]
```

B [1] Die Werte SYS1, SYS2, SYS3 und SYS4 sind nur in BS2000/OSD erlaubt.

B [2] *ANY ist nur in BS2000/OSD erlaubt.

B B B B ANNOAMSG=(**anno**unce **a**synchronous **m**e**s**sa**g**e)

Gilt nur für LTERM-Pools, über die sich Terminals an die UTM-Anwendung anschließen. Legt fest, ob openUTM asynchrone Nachrichten vor der Ausgabe in der Systemzeile am Terminal ankündigt.

B B Y

Asynchrone Nachrichten sollen angekündigt werden. Der Benutzer muss die Nachricht mit dem Kommando KDCOUT anfordern.

B

Standard: Y

B B B B N

Asynchrone Nachrichten werden ohne Ankündigung gesendet.
Wenn ANNOAMSG=N generiert ist, kann zu den Terminals dieses LTERM-Pools erst ab OMNIS V7.0 ein Verbindungsaufbau über eine Multiplexverbindung (siehe MUX-Anweisung auf Seite 417) erfolgen.

BCAMAPPL=local_appliname

Name der lokalen UTM-Anwendung, über den die Verbindung zwischen Client und UTM-Anwendung aufgebaut wird. *local_appliname* wird entweder mit MAX ...,APPLINAME= oder in der BCAMAPPL-Anweisung festgelegt, siehe hierzu auch die Beschreibung der BCAMAPPL-Anweisung auf Seite 290.

Wird beim Operanden PTYPE= ein Wert ungleich APPLI, SOCKET oder UPIC-R angegeben, dann kann für *local_appliname* nur der mit MAX ...,APPLINAME=*appliname* definierte Name angegeben werden.

Der in der CLUSTER-Anweisung angegebene BCAMAPPL-Name ist hier nicht erlaubt.

Standard: *appliname*, der bei MAX ...,APPLINAME= angegeben wurde.

B *BS2000/OSD:*

B B B B In der BCAMAPPL-Anweisung können Sie festlegen, ob bei der Kommunikation mit Kommunikationspartnern, die sich an diese Anwendung anschließen, NEA-, ISO-Transportprotokolle oder native TCP/IP (Socket-Schnittstelle) verwendet werden sollen.

B B Der Client muss i.A., wenn er eine Verbindung zu der UTM-Anwendung aufbauen will, *local_appliname* als Partnernamen angeben.

B B B B Eine Ausnahme bilden LTERM-Pools, die mit PTYPE=SOCKET generiert sind. In diesem Fall müssen die Clients, die sich über den LTERM-Pool anschließen, die Portnummer kennen, an der die UTM-Anwendung „horcht“. Diese Portnummer wird in BCAMAPPL LISTENER-PORT= festgelegt.

CONNECT-MODE=

legt fest, ob ein Client sich mehrfach unter demselben Namen über den LTERM-Pool an die UTM-Anwendung anschließen kann.

SINGLE Jeder Client kann sich nur einmal unter demselben Namen an den LTERM-Pool anschließen.

Standard: SINGLE

MULTI Nur zulässig für LTERM-Pools, über die sich UPIC-Partner oder TS-Anwendungen anschließen.
Eine mehrfach auf demselben Rechner gestartete UPIC-Client-Anwendung (PTYPE=UPIC-R oder UPIC-L) bzw. TS-Anwendung (PTYPE=APPLI oder SOCKET) kann sich mehrfach unter demselben Namen über den LTERM-Pool an die UTM-Anwendung anschließen. Es muss nicht für jede Verbindung ein neuer Name erzeugt werden.

Eine UPIC-Client- bzw. TS-Anwendung kann sich maximal *number1*-mal an den LTERM-Pool anschließen (siehe NUMBER=*number1*, Seite 530).

Im Fall CONNECT-MODE=MULTI identifiziert die UTM-Anwendung den Kommunikationspartner bzw. die Verbindung zum Partner nicht (wie gewöhnlich) über den Partner-Namen, den der Partner beim Verbindungsaufbau angibt. Der Partner ist der UTM-Anwendung unter seinem Anwendungsnamen nicht bekannt. Die Identifikation erfolgt stattdessen über den Namen des Pool-LTERM-Partners (*ltermname*), über den er angeschlossen ist. Damit openUTM den Partner eindeutig identifizieren kann, darf das Tripel aus *ltermname* des LTERM-Pools, *processorname* und *local_appliname* in keiner PTERM-, CON- oder OSI-CON-Anweisung explizit generiert sein. Darüber hinaus darf der Name, den der Partner beim Verbindungsaufbauwunsch angibt, mit keinem LTERM-Namen des LTERM-Pools übereinstimmen.

ENCRYPTION-LEVEL=

Nur relevant für UPIC-Clients (≥ V5.0), die die Verschlüsselung unterstützen, und unter BS2000/OSD zusätzlich für einige Terminalemulationen, die die Verschlüsselung unterstützen.

In ENCRYPTION-LEVEL legen Sie die minimale Verschlüsselungsebene für die Kommunikation mit Clients fest, die sich über diesen LTERM-Pool an die Anwendung anschließen.
Sie geben an, ob die UTM-Anwendung für Verbindungen über den LTERM-Pool die Verschlüsselung der Nachrichten fordern soll oder nicht. Sie können den LTERM-Pool auch als „trusted“ definieren, d.h. alle Clients, die sich über den LTERM-Pool anmelden, werden von der Anwendung wie „trusted Clients“ behandelt (zur Verschlüsselung siehe auch Abschnitt „Nachrichten-Verschlüsselung auf Verbindungen zu Clients“ auf Seite 225).

Folgende Angaben sind möglich:

NONE Die Verschlüsselung der Nachrichten, die zwischen Client und UTM-Anwendung ausgetauscht werden, wird von openUTM **nicht** standardmäßig angefordert.
Passworte werden verschlüsselt übertragen, sofern beide Partner Verschlüsselung unterstützen, und zwar mit dem größten verfügbaren Schlüssel, d.h. mit AES, falls ein RSA-Schlüssel mit Länge ≥ 512 bit verfügbar ist, sonst mit DES. Der AES-Schlüssel wird wiederum mit dem höchsten verfügbaren RSA-Schlüssel verschlüsselt.
Vorgänge, für deren Vorgangs-TACs Verschlüsselung generiert wurde (siehe ENCRYPTION-LEVEL in der TAC-Anweisung auf Seite 495), können über diesen LTERM-Pool nur gestartet werden, wenn der angeschlossene Client beim Conversation- bzw. Verbindungsaufbau explizit eine Verschlüsselungsebene auswählt, die mindestens der benötigten Ebene entspricht.

Standard: NONE

1 | 2 | 3 | 4 Nachrichten, die zwischen dem Client und der UTM-Anwendung ausgetauscht werden, werden von openUTM standardmäßig verschlüsselt. Der Wert (1 ... 4) gibt die Verschlüsselungsebene an. Es können sich nur Clients über diesen LTERM-Pool anschließen, die mindestens diese Verschlüsselungsebene unterstützen. Unterstützt ein Client die angegebene Verschlüsselungsebene nicht, lehnt openUTM den Verbindungsaufbau zum Client ab.

Die Werte 1 bis 4 haben folgende Bedeutung:

1 Passworte und Ein-/Ausgabe-Nachrichten werden mit dem DES-Algorithmus verschlüsselt. Zum Austausch des DES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 200 bit verwendet.

2 Passworte und Ein-/Ausgabe-Nachrichten werden mit dem AES-Algorithmus verschlüsselt. Zum Austausch des AES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 512 bit verwendet.

3 Passworte und Ein-/Ausgabe-Nachrichten werden mit dem AES-Algorithmus verschlüsselt. Zum Austausch des AES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 1024 bit verwendet.

4 Passworte und Ein-/Ausgabe-Nachrichten werden mit dem AES-Algorithmus verschlüsselt. Zum Austausch des AES-Schlüssels wird ein RSA-Schlüssel mit einer Schlüssellänge von 2048 bit verwendet.

ENCRYPTION-LEVEL=1 ... 4 ist für UPIC-Partner nur sinnvoll, wenn an Ihrem System die Encryption-Funktionalität von openUTM installiert ist. Andernfalls kann sich über den LTERM-Pool kein Client anschließen.

Für VTSU-Partner wird die Verschlüsselung von VTSU durchgeführt.

Für die einzelnen Client-Typen gilt bezüglich der Verschlüsselungsebene:

– Für remote UPIC-Clients (PTYPE=UPIC-R) sind die Verschlüsselungsebenen 1 bis 4 sinnvoll.

B B B – Für Clients mit PTYPE= T9763 oder *ANY unter BS2000/OSD ist nur die Verschlüsselungsebene 1 sinnvoll (ENCRYPTION-LEVEL=1). Die Werte 2, 3 und 4 werden von KDCDEF ohne Meldung in 1 geändert.

X/W X/W X/W – Bei lokalen UPIC-Clients (PTYPE=UPIC-L) einer Anwendung unter Unix- bzw. Windows-Systemen wird 1, 2, 3, oder 4 ohne Meldung in TRUSTED geändert.

– Wird 1 ... 4 für einen Partner eines anderen Typs angegeben, dann wird der Wert von openUTM ohne Meldung in NONE geändert.

Falls die Anwendung mit OPTION GEN-RSA-KEYS=NO generiert ist, dann werden beim KDCDEF-Lauf keine RSA-Schlüssel erzeugt. Um die Verschlüsselungsfunktionen nutzen zu können, müssen Sie die benötigten Schlüssel per Administration erzeugen (KC_ENCRYPT oder WinAdmin) oder per KDCUPD aus einer alten KDCFILE übertragen.

TRUSTED Nachrichten zwischen Client und Anwendung werden nicht verschlüsselt. Ein Client, der sich über diesen LTERM-Pool an die Anwendung anschließt, kann aber trotzdem Vorgänge starten, deren Vorgangs-TACs Verschlüsselung erfordern (generiert mit TAC ENCRYPTION-LEVEL=1 oder 2). D.h. jeder Client, der sich über diesen LTERM-Pool anschließt, wird als „trusted Client“ betrachtet.

TRUSTED sollten Sie nur dann für einen LTERM-Pool generieren, wenn alle zum Pool gehörenden „physikalischen“ Clients nicht für jedermann zugänglich sind und die Kommunikation mit den Clients über eine sichere Verbindung läuft.

X/W X/W X/W X/W

Unix-, Windows-Systeme:
Für lokale UPIC-Clients (UPIC-L) ist nur der Wert TRUSTED erlaubt. Andere Angaben werden von openUTM ignoriert und automatisch auf TRUSTED gesetzt.

B B B B B FORMAT= definiert das Startformat für Benutzer an Terminals, die über diesen LTERM-Pool an die Anwendung angemeldet wurden (siehe dazu auch Anweisung LTERM ...,FORMAT=, Seite 358). Nach dem Aufbau der Verbindung wird das unter *formatname* beschriebene Format am Terminal ausgegeben, sofern kein Terminal-spezifischer Wiederanlauf durchgeführt wird.

B Standard: Kein Startformat.

IDLETIME=time

legt die Zeit in Sekunden fest, die openUTM außerhalb einer Transaktion, d.h. nach dem Ende einer Transaktion oder nach dem Anmelden, maximal auf eine Eingabe vom Client wartet. Bei Zeitüberschreitung baut openUTM die Verbindung zum Client ab. Ist der Client ein Terminal, dann wird vor dem Verbindungsabbau die Meldung K021 ausgegeben.

Der für *time* angegebene Wert darf nicht kleiner sein als die Timer TERMWAIT=*time* (Wartezeit zwischen Dialog-Ausgabe und -Antwort innerhalb einer Mehrschritt-Transaktion; siehe Seite 401) und PGWTTIME=*time* (Zeit, die ein Teilprogramm nach einem blockierenden Aufruf auf das Eintreffen einer Nachricht wartet; siehe Seite 392), die Sie in der MAX-Anweisung festlegen.

Diese Funktion dient dazu den Datenschutz zu verbessern:
Vergisst ein Benutzer bei einer Unterbrechung oder der Beendigung seiner Arbeit am Terminal, sich abzumelden, dann wird die Verbindung zum Terminal bzw. zum Client nach Ablauf der Wartezeit automatisch abgebaut. Damit wird die Wahrscheinlichkeit für einen unberechtigten Zugang verringert.

Standard: 0 (= Warten ohne Zeitbegrenzung)
Für Terminals wird (falls gesetzt) MAX TERMWAIT=(...,*time2*) gesetzt.
Maximalwert: 32767
Minimalwert: 60

Geben Sie einen Wert an, der kleiner als der Minimalwert ist, dann ersetzt KDCDEF den Wert durch den Minimalwert.

B KERBEROS-DIALOG =

B B B Y Beim Verbindungsaufbau wird für Terminals, die Kerberos unterstützen und die sich direkt (nicht über OMNIS) über diesen Terminalpool an die Anwendung anschließen, ein Kerberos-Dialog durchgeführt.

B B B B B B B B openUTM speichert die Kerberos-Information in der Länge ab, die sich aus dem Maximum der bei MAX PRINCIPAL-LTH und MAX CARDLTH generierten Längen ergibt. Wenn die Kerberos-Information länger ist, wird sie auf diese Länge verkürzt abgespeichert.
Wenn weder bei MAX PRINCIPAL-LTH noch bei MAX CARDLTH eine Länge größer Null generiert ist, dann wird eine Warnungsmeldung ausgegeben.

B B B B B Mit dem KDCS-Aufruf INFO (KCOM=CD) kann ein Teilprogrammlauf diese Information lesen.
Ausnahme: An diesem Client hat sich anschließend ein Benutzer mit Ausweiskarte angemeldet. In diesem Fall wird die Kerberos-Information durch die Ausweis-Information überschrieben.

B N Es wird kein Kerberos-Dialog durchgeführt.

B Standard.

KSET=keysetname1

Name eines Keysets, das diesem LTERM-Pool zugeordnet wird. Das Keyset muss mit einer KSET-Anweisung definiert werden.

Es legt die Zugriffsrechte der LTERM-Partner dieses LTERM-Pools für die Nutzung von Services der Anwendung und von entfernten Services (LTAC) fest, die in dieser Anwendung generiert wurden.

Über einen LTERM-Partner dieses LTERM-Pools können mit Lockcode bzw. Access-List geschützte Services der Anwendung nur gestartet bzw. mit Lockcode bzw. Access-List geschützte entfernte Services nur adressiert werden, wenn Folgendes zutrifft: Sowohl im zugeordneten Keyset des LTERM-Partners als auch im KSET der UTM-Benutzerkennung, unter der die Anmeldung über diesen LTERM-Partner erfolgte, ist der zum Lockcode bzw. der Access-List passende Key- bzw. Zugangscode enthalten.

Bei PTYPE=APPLI, SOCKET, UPIC-R, UPIC-L gilt bezüglich des Keysets der Benutzerkennung zusätzlich Folgendes:

– Übergibt der Client für die Session/Conversation keine echte Benutzerkennung an openUTM, dann ergeben sich seine Zugriffsrechte aus der Menge der Keycodes, die sowohl in dem mit KSET als auch in dem mit USER-KSET generierten Keyset vorhanden sind.
Das Keyset *keysetname1* sollte deshalb alle Keycodes enthalten, die auch in dem mit USER-KSET generierten Keyset enthalten sind.

– Übergibt der Client eine Benutzerkennung, dann ergeben sich die Zugriffsrechte aus der Menge der Keycodes, die sowohl in dem Keyset der Benutzerkennung als auch in dem mit KSET generierten Keyset enthalten sind.

B LOCALE=(lang_id,terr_id,ccsname)

B definiert die Sprachumgebung der Clients, die sich über den LTERM-Pool

B an die UTM-Anwendung anschließen.

B lang_id Max. 2 Zeichen langes Sprachkennzeichen für die Clients des LTERM-

B Pools. Das Sprachkennzeichen ist frei wählbar.

B Das Sprachkennzeichen kann von den Teilprogrammen der Anwendung

B abgefragt werden, so dass es Nachrichten an die Terminals in der Landessprache des Clients übertragen kann.

B terr_id Max. 2 Zeichen langes Territorialkennzeichen der Clients des LTERM-

B Pools. Das Territorialkennzeichen ist frei wählbar.

Das Territorialkennzeichen kann von den Teilprogrammen der Anwendung abgefragt werden. So können in Nachrichten die territorialen Besonderheiten in der Landessprache des Clients berücksichtigt werden.

ccsname (**c**oded **c**haracter **s**et **name**)
Für *ccsname* ist der max. 8 Zeichen lange Name eines erweiterten Zeichensatzes (CCS-Name) anzugeben. Der angegebene CCS-Name muss zu einem im BS2000-System definierten EBCDIC-Zeichensatz gehören (siehe auch Benutzerhandbuch zu XHCS). Der Zeichensatz muss kompatibel sein zu einem ISO-Zeichensatz, der von allen zum LTERM-Pool gehörenden Terminals unterstützt wird.
Zum Zeitpunkt der Generierung kann KDCDEF weder die Gültigkeit des CCS-Namens im BS2000-System noch die Kompatibilitätsbedingung überprüfen. Der zum CCS-Namen gehörende Zeichensatz wird verwendet für:

- die Ausgabe von Dialog-Nachrichten an 8-Bit-Terminals, wenn die Anwendung ohne Benutzerkennungen generiert ist, bzw. noch kein Benutzer am LTERM-Partner des LTERM-Pools angemeldet ist, und wenn nicht explizit ein anderer CCS-Name über Editprofil oder Format ausgewählt wurde.
- die Ausgabe asynchroner Nachrichten an 8-Bit-Terminals, wenn nicht explizit ein anderer CCS-Name über Editprofil oder Format ausgewählt wurde.

Standard: Wird TPOOL ...,LOCALE nicht angegeben, dann wird das in der MAX-Anweisung definierte Locale der Anwendung verwendet.

LOCK=lockcode

Zugriffsschutz des LTERM-Pools. Lockcode, der den LTERM-Partnern des LTERM-Pools zugeordnet wird. *lockcode* ist ein Zahlenwert zwischen 1 und dem in der Anwendung erlaubten Maximalwert (MAX ...,KEYVALUE=). An einen LTERM-Partner dieses LTERM-Pools können Sie sich an die Anwendung nur unter einer UTM-Benutzerkennung (USER) anmelden, für die ein Keyset mit einem Keycode generiert wurde, der mit dem Lockcode des LTERM-Pools übereinstimmt.

Standard: 0 (LTERM-Pool ist nicht durch Lockcode geschützt)
Maximalwert: Wert von KEYVALUE aus der MAX-Anweisung

LTERM=ltermprefix

Präfix für die Namen der LTERM-Partner des LTERM-Pools. Die LTERM-Namen sind 8 Zeichen lang und setzen sich zusammen aus dem hier angegebenen Präfix und einer Laufnummer. Die Laufnummer geht von 1 bis zu dem Wert, den Sie mit NUMBER=*number1* angeben.

Die maximale Länge von *ltermprefix* hängt davon ab, wieviele Dezimalstellen *number1* besitzt. D.h. die Anzahl der Zeichen von *ltermprefix* plus die Anzahl der Dezimalstellen von *number1* darf nicht größer als 8 sein.

Bei der Angabe von *ltermprefix* und *number1* ist zu beachten, dass die Namen der LTERM-Partner innerhalb einer Anwendung eindeutig sein müssen. Dies gilt für die mit TPOOL ...,LTERM= erzeugten Namen (in allen TPOOL-Anweisungen) und die mit LTERM-Anweisungen fest definierten Namen.

Beispiel
Ist *number1*=1000 und LTERM=LTRM, dann heißen die für den LTERM-Pool definierten LTERM-Partner LTRM0001, LTRM0002, ..., LTRM1000.

Diese Namen dürfen in keiner LTERM-Anweisung angegeben werden!

Zusätzlich dürfen die LTERM-Namen keinem weiteren Objekt der Namensklasse 1 zugeordnet sein. Siehe dazu auchAbschnitt „Eindeutigkeit der Namen und Adressen“ auf Seite 262.

MAP= In MAP legen Sie fest, ob openUTM die Benutzer-Nachrichten, die mit dem Kommunikationspartner ausgetauscht werden, umcodieren soll oder nicht (EBCDIC <-> ASCII).
Benutzer-Nachrichten werden an der KDCS-Schnittstelle bei den Aufrufen zur Nachrichtenbehandlung (MPUT/MGET/FPUT/DPUT/FGET) im Nachrichtenbereich übergeben.

MAP= regelt die Konvertierung beim Austausch unformatierter Nachrichtern. Bei formatierten Nachrichten findet grundsätzlich keine Nachrichtenbehandlug statt.

B *BS2000/OSD:*

B MAP≠USER ist nur für LTERM-Pools mit PTYPE=SOCKET erlaubt.

X/W *Unix-, Windows-Systeme:*

X/W MAP≠USER ist bei allen TS-Anwendungen (PTYPE=APPLI oder SOCKET) erlaubt.
X/W

USER openUTM konvertiert die Daten des Nachrichtenbereichs nicht, d.h. die Nachrichten werden unverändert an die Partner-Anwendung übertragen bzw. die vom Partner empfangenen Nachrichten werden unverändert an das Teilprogramm übergeben.
Dabei ist zu beachten, dass der Transaktionscode in der Benutzer-Nachricht enthalten ist. Er muss so kodiert sein, wie ihn die UTM-Anwendung erwartet, d.h. unter BS2000/OSD in EBCDIC und unter Unix-/Windows-Systemen in ASCII.

Standard: USER

B SYSTEM | SYS | SYS1 | SYS2 | SYS3 | SYS4 (BS2000/OSD)
B muss nur angegeben werden, wenn die vom Kommunikationspartner empfangenen Nachrichten nicht in EBCDIC codiert sind, bzw. wenn der Kommunikationspartner in ASCII codierte Nachrichten von der UTM-Anwendung erwartet.

B Geben Sie einen der obigen Werte an, dann konvertiert openUTM die Daten der Nachrichten vor dem Senden von EBCDIC nach ASCII und nach dem Empfangen von ASCII nach EBCDIC. Dabei geht openUTM davon aus, dass die Nachrichten nur abdruckbare Zeichen enthalten.

B Mit SYSTEM, SYS, SYS1, SYS2, SYS3, SYS4 geben Sie die Umsetztabelle an, die openUTM bei der Umcodierung verwenden soll.
Die Umsetztabellen müssen in dem Modul KDCEA definiert sein, siehe Seite 609.

B Die Angaben SYSTEM, SYS und SYS1 sind dabei synonym. Geben Sie einen dieser Werte an, dann verwendet openUTM für die Konvertierung die Standard-Codetabelle, die von EBCDIC nach 7-Bit ASCII umsetzt. Die Standard-Codetabelle ist in KDCEA bereits vordefiniert und kann ohne weitere Vorbereitungen genutzt werden.

B Soll openUTM eine andere Umsetzung von EBCDIC nach ASCII durchführen, dann müssen Sie in KDCEA entsprechende Umsetztabellen selbst definieren. Sie können in KDCEA maximal drei eigene Umsetztabellen definieren (Tabelle 2 bis Tabelle 4).

B Geben Sie SYS2 an, dann konvertiert openUTM die Benutzer-Nachrichten mit Hilfe der Tabelle 2. Bei SYS3 verwendet openUTM die Tabelle 3 und bei SYS4 die Tabelle 4.

X/W SYSTEM Unix-, Windows-Systeme)
X/W openUTM konvertiert die Daten des Nachrichtenbereichs vor dem Absenden von ASCII nach EBCDIC bzw. nach dem Empfangen von EBCDIC nach ASCII. Die Nachricht darf dabei nur abdruckbare Zeichen enthalten und muss im Zeilenmodus (KCMF = Leerzeichen) erzeugt werden. Der Parameter ist nur für PTYPE=APPLI oder SOCKET zulässig.

B NETPRIO= bezeichnet die Transportpriorität, die auf den diesem LTERM-Pool zugeordneten Transportverbindungen benutzt werden soll.

B NETPRIO hat keine Bedeutung, wenn die Verbindung von der Partner-Anwendung über die Socket-Schnittstelle aufgebaut wird (PTYPE=SOCKET).

B Standard: MEDIUM

NUMBER=number1

number1 definiert die Anzahl der LTERM-Partner für diesen LTERM-Pool, d.h. es können sich maximal *number1* Clients über den LTERM-Pool an die Anwendung anschließen. Der zulässige Maximalwert für *number1* hängt von der Anzahl der Namen ab, die in der UTM-Anwendung generiert wurden (siehe Abschnitt „Anzahl der Namen“ auf Seite 259).

Minimalwert: 1

PRONAM= gibt an, auf welchem Rechner sich die Clients befinden müssen, um sich über diesen LTERM-Pool an die Anwendung anschließen zu können.

B *BS2000/OSD*:

B PRONAM= ist Pflichtoperand.

B Wird PROTOCOL=STATION gesetzt, muss die Kombination aus
B PRONAM / PTYPE / BCAMAPPL eindeutig sein.

B Wird PROTOCOL=NO gesetzt, muss die Kombination aus PRONAM und
B BCAMAPPL eindeutig sein.

X/W *Unix-, Windows-Systeme*:

X/W PRONAM= darf nur für LTERM-Pools vom Typ PTYPE=APPLI, SOCKET
X/W oder UPIC-R angegeben werden.

X/W Die Kombination aus PRONAM / PTYPE / BCAMAPPL muss eindeutig
X/W sein.

X/W Standard in Unix- und Windows-Systemen: 8 Leerzeichen

{ processorname | C’processorname’ }

Maximal 8 Zeichen langer symbolischer Rechnername. Nur Clients, die an diesen Rechner angeschlossen sind, können sich über diesen LTERM-Pool an die Anwendung anmelden.
Enthält *processorname* Sonderzeichen, muss er als Zeichenkette mit C’...’ angegeben werden.

Wird ein LTERM-Pool mit PTYPE= SOCKET generiert, dann müssen Sie für *processorname* die symbolische Adresse (offiziellen Hostnamen) des Partner-Rechners angeben, unter dem die IP-Adresse des Partner-Rechners im Name Service des lokalen Systems (z.B. die hosts-Datei in Unix-/Windows-Systemen oder die RDF-Datei in BS2000/OSD) eingetragen ist. Einen Aliasnamen des Rechners dürfen Sie nicht angeben.

X/W X/W *Unix-, Windows-Systeme:*
Für die Angabe von *processorname* haben Sie zwei Möglichkeiten:

X/W X/W X/W X/W – Sie geben den realen Hostnamen an, unter dem die IP-Adresse des Partner-Rechners im Name Service des lokalen Systems (z.B. der hosts-Datei) eingetragen ist. Einen Aliasnamen des Rechners dürfen Sie nicht angeben.

X/W X/W X/W X/W X/W – Sie geben den UTM-Hostnamen des Partner-Rechners an. Voraussetzung hierfür ist, dass Sie die Umgebungsvariable UTM_NET_HOSTNAME gesetzt und den UTM-Hostnamen in der Umwandlungsdatei festgelegt haben (siehe Abschnitt „Mapped Hostnamen verwenden (Unix- und Windows-Systeme)“ auf Seite 118).

*ANY Über den LTERM-Pool kann sich jeder Client an die Anwendung anmelden, der die folgenden Bedingungen erfüllt:

– Der Client darf nicht explizit mit einer PTERM-Anweisung generiert sein
– Der Typ des Clients muss mit der Angabe in PTYPE übereinstimmen
– Für den Rechner, auf dem der Client residiert, und den Typ des Clients darf kein anderer LTERM-Pool generiert sein. Damit soll verhindert werden, dass offene LTERM-Pools als „Überlaufbecken“ für andere LTERM-Pools genutzt werden.

B B B PROTOCOL= gibt an, ob zwischen der UTM-Anwendung und den über diesen LTERM-Pool angeschlossenen Clients das Benutzerdienstprotokoll (NEABT) gefahren werden soll.

B B B B B B B B N Es wird durch openUTM kein Benutzerdienstprotokoll gefahren.
Wenn PROTOCOL=N generiert ist, kann zu den LTERM-Partnern dieses LTERM-Pools kein Verbindungsaufbau über eine Multiplexverbindung (siehe MUX-Anweisung auf Seite 417) erfolgen.
Für UPIC-Clients (PTYPE=UPIC-R) und TS-Anwendungen (PTYPE=APPLI/SOCKET) darf nur PROTOCOL=N generiert werden. Die Angabe PROTOCOL=STATION wird in diesem Fall ohne Meldung ignoriert.

B B Haben Sie PTYPE=*ANY angegeben, dann ignoriert openUTM die Angabe PROTOCOL=NO.

B B STATION Zwischen der UTM-Anwendung und den über diesen LTERM-Pool angeschlossenen Clients wird das Benutzerdienstprotokoll (NEABT) gefahren.

B B B B Bei PTYPE=*ANY muss PROTOCOL=STATION angegeben werden. openUTM benötigt in diesem Fall das Benutzerdienstprotokoll (NEABT) zur Bestimmung des Partnertyps, wenn der Typ bei der Generierung nicht explizit angeben wurde (PTYPE=*ANY).

B B Standard:
N bei PTYPE=APPLI, SOCKET oder UPIC-R

B STATION bei PTYPE≠APPLI, SOCKET oder UPIC-R.

PTYPE= Typ des Clients, der sich über diesen LTERM-Pool an die Anwendung anschließen kann.

Haben Sie in BCAMAPPL= einen Anwendungsnamen angegeben, der für die Kommunikation über die Socket-Schnittstelle generiert ist (BCAMAPPL-Anweisung mit T-PROT=SOCKET), dann müssen Sie PTYPE=SOCKET setzen.

Pflichtoperand

partnertyp gibt explizit den Typ des Clients an.
Eine Liste der unterstützten Partnertypen finden Sie bei der Beschreibung der Steueranweisung PTERM auf Seite 454f (BS2000/OSD) und auf Seite 459 (Unix-, Windows-Systeme). Beachten Sie, dass über LTERM-Pools **keine** Drucker angeschlossen werden können!

B *ANY Nur unter BS2000/OSD erlaubt.
B PTYPE=*ANY beschreibt einen offenen LTERM-Pool. An diesen LTERM-
B Pool können sich alle Clients anschließen, die das Benutzerdienstprotokoll
B unterstützen (PROTOCOL=STATION) und die sich auf dem Rechner befin-
B den, der mit PRONAM= festgelegt wurde.

B In diesem Fall nimmt openUTM den Typ des Partners beim Verbindungs-
B aufbau aus dem Benutzerdienstprotokoll. Erst dann wird entschieden, ob
B der Partnertyp unterstützt wird.

B Der Vorteil von PTYPE=*ANY ist, dass Sie Clients in die Konfiguration auf-
B nehmen können, ohne zu wissen, wie sie im PDN generiert sind. Darüber
B hinaus wird die Pflege der Konfiguration erleichtert, denn auch nach einer
B Umgenerierung im PDN können diese Clients weiterhin eine Verbindung zu
B der Anwendung aufbauen, ohne dass Sie die KDCDEF-Generierung än-
B dern müssen.

QLEV=queue_level_number
(**q**ueue **lev**el)
gibt an, wieviele asynchrone Nachrichten in der Message Queue des LTERM-Partners maximal zwischengespeichert werden können. Wird der Schwellwert überschritten, so werden weitere FPUTs an diesen LTERM-Partner mit der Meldung 40Z abgewiesen.

Standard: 32767
Minimalwert: 0
Maximalwert: 32767

Wird ein Wert angegeben, der größer als der Maximalwert ist, dann wird im KDCDEF-Lauf ohne Meldung der Standardwert gesetzt.

STATUS= Mit NUMBER=*number1* legen Sie die Anzahl der LTERM-Partner für den LTERM-Pool fest. Mit STATUS=*number2* geben Sie an, wieviele Clients beim Start der Anwendung für den LTERM-Pool zugelassen (ON) bzw. gesperrt (OFF) sind. Während des Betriebs der Anwendung kann *number2* per Administration geändert werden.

Standard: STATUS=(OFF , 0), d.h. alle Clients des LTERM-Pools sind zugelassen.

ON Es werden *number2* Clients zugelassen.

OFF Es werden *number2* Clients gesperrt.

number2 Anzahl der Clients, und damit Anzahl der LTERM-Partner des LTERM-Pools, die beim Anwendungsstart zugelassen oder gesperrt sind.

TERMN=termn_id

Kennzeichen für die Art des Clients, das openUTM dem Anwendungsprogramm im Feld KCTERMN des KB-Kopfes zur Verfügung stellt. *termn_id* ist max. 2 Zeichen lang.

termn_id wird nicht von openUTM abgefragt, es kann vom Benutzer zur Auswertung verwendet werden.

Standardwerte:

Wird der Operand nicht angegeben, so trägt openUTM in KCTERMN das Standard-Kennzeichen für den Partnertyp ein, der beim Operanden PTYPE angegeben wird. Der Benutzer kann jedoch auch andere Werte wählen.

B – *BS2000/OSD:*
B Die Standardwerte sind der Partnertyp-Tabelle der PTERM-Anweisung, Operand PTYPE= auf Seite 454 zu entnehmen.
B Wird TERMN bei Clients, die mit PTYPE=*ANY generiert sind, nicht explizit angegeben, dann trägt openUTM das Kennzeichen für die Terminalmnemonic erst beim Aufbau der Verbindung in KCTERMN ein. Es
B wird die Standard-Terminalmnemonic des Typs eingetragen, der im Benutzerdienstprotokoll der Verbindungsanforderung enthalten ist.
B
B
B
B
B

X/W
X/W

– *Unix-, Windows-Systeme*:
Die Standardwerte sind der folgenden Tabelle zu entnehmen.

X/W	**PTYPE**	**TERMN**
X/W	TTY	F1
X/W	APPLI	A1
X/W	UPIC-L	A2
X/W	UPIC-R	A5
X/W	SOCKET	A7

USER-KSET=keysetname2

nur erlaubt, wenn die Anwendung mit Benutzerkennungen generiert wird und PTYPE=APPLI, SOCKET, UPIC-R oder UPIC-L angegeben wird. USER-KSET= dürfen Sie nur zusammen mit KSET= setzen.

Für *ksetname2* ist der Name eines Keyset anzugeben. Das Keyset muss mit einer KSET-Anweisung definiert werden.

Mit USER-KSET= legen Sie die minimalen Zugriffsrechte fest, die ein über diesen LTERM-Pool verbundener Client ausüben kann.

ksetname2 wird wirksam, wenn der Client unter der Verbindungs-Benutzerkennung angemeldet ist. Seine Zugriffsrechte ergeben sich aus der Menge der Keycodes, die sowohl in dem mit KSET= als auch in dem mit USER-KSET= generierten Keyset enthalten sind (Schnittmenge). Aus diesem Grund sollten alle Keycodes, die USER-KSET=*ksetname2* enthält, auch in KSET=*ksetname1* enthalten sein.

Standard: kein Keyset
es gelten immer die in KSET festgelegten Zugriffsrechte

USP-HDR= Legt fest, für welche Ausgabe-Nachrichten openUTM auf den mit dieser Anweisung generierten Verbindungen einen UTM-Socket-Protokoll-Header aufbauen soll.

Ein Wert ungleich NO darf nur bei LTERM-Pools angegeben werden, die für die Kommunikation über Socket-Verbindungen konfiguriert sind (PTYPE=SOCKET).

Eine Beschreibung des USP-Headers finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“.

ALL Bei allen Ausgabe-Nachrichten (Dialog, Asynchron, K-Meldungen) erzeugt openUTM einen UTM-Socket-Protokoll-Header und stellt diesen der Nachricht voran.

MSG Nur bei der Ausgabe von K-Meldungen erzeugt openUTM einen UTM-Socket-Protokoll-Header und stellt diesen der Nachricht voran.

NO Es werden keine UTM-Socket-Protokoll-Header erzeugt.

Standard: NO

TRANSFER-SYNTAX - Transfersyntax definieren

Die Steueranweisung TRANSFER-SYNTAX benötigen Sie nur, wenn Sie für die Kommunikation über das OSI TP-Protokoll einen eigenen Application Context definieren wollen (siehe Anweisung APPLICATION-CONTEXT auf Seite 283).

Die Anweisung definiert für eine Transfersyntax einen lokalen Namen und weist diesem einen Object-Identifier zu. Die Transfersyntax legt fest, mit welcher Regel eine über den Object Identifier identifizierte Abstrakte Syntax encodiert bzw. decodiert wird.

Die hier definierte Transfersyntax muss von der eingesetzten OSS-Version unterstützt werden. Zur Zeit unterstützt OSS ausschließlich die Transfersyntax BER.

```
TRANSFER-SYNTAX␣ transfer_syntax_name

                  ,OBJECT-IDENTIFIER=object_identifier
```

transfer_syntax_name
: Ein max. 8 Zeichen langer Name, der lokal für eine Transfersyntax vergeben wird. Innerhalb einer UTM-Anwendung muss jede Transfersyntax eindeutig sein.

  Der *transfer_syntax_name* BER (Basic Encoding Rules) ist reserviert.

OBJECT-IDENTIFIER=object_identifier
: Object-Identifier der Transfersyntax, der wie folgt angegeben wird:

  object_identifier=(number1,number2, ... ,number10)

  number ist dabei eine positive ganze Zahl im Bereich von 0 bis 67108863. Für *object_identifier* werden in Klammern und durch Komma getrennt mindestens 2 bis maximal 10 Zahlen angegeben. Die Anzahl und Position der Zahlen ist relevant.
  Anstelle der Zahl kann auch der dieser Zahl zugeordnete symbolische Name angegeben werden. Der Tabelle auf Seite 92 können Sie entnehmen, welchen Wert *number* an dieser Position annehmen darf.

  object_identifier muss innerhalb der UTM-Anwendung eindeutig sein, d.h. es dürfen keine weiteren Transfersyntaxen mit demselben Object-Identifier generiert werden.

openUTM generiert standardmäßig die Transfersyntax BER:

```
TRANSFER-SYNTAX␣BER,                                          -
               OBJECT-IDENTIFIER=(2, 1, 1)                    -
```

Symbolische Beschreibung des Object-Identifiers:

```
(joint-iso-ccitt, ansl, basic-encoding)
```

ULS - Namen von ULS festlegen

Jeder UTM-Benutzerkennung kann ein User-spezifischer Langzeitspeicher (ULS) zugeordnet werden, der wiederum aus mehreren Blöcken bestehen kann. Jeder Block wird über einen Namen angesprochen.

In UTM-Cluster-Anwendungen, die mit CLUSTER GLOBAL-UTM-DATA=NO (Standard) generiert sind, dürfen Sie keine ULS-Bereiche generieren.

Mit der Steueranweisung ULS wird der Name eines ULS-Blockes festgelegt. openUTM stellt damit jeder UTM-Benutzerkennung einen ULS-Block mit diesem Namen zur Verfügung. Durch Wiederholen der ULS-Anweisung mit unterschiedlichen Blocknamen können mehrere Blöcke eingerichtet werden.

Die mit der ULS-Anweisung vereinbarten ULS-Blöcke werden auch den Sessions (LSES) bei verteilter Verarbeitung über LU6.1 zugeordnet.

Die Anweisung darf nur angegeben werden, wenn die Anwendung mit Benutzerkennungen generiert wird.

Es sind maximal 100 ULS-Anweisungen zugelassen.

```
ULS␣          blockname
```

blockname Max. 8 Zeichen langer Blockname. Mit *blockname* kann der ULS-Block aus einem Teilprogramm heraus angesprochen werden.

USER - Benutzerkennungen definieren

Mit der Steueranweisung USER definieren Sie Benutzerkennungen für die UTM-Anwendung. Über UTM-Benutzerkennungen können sich Benutzer und Client-Programme an die UTM-Anwendung anmelden. Für Benutzerkennungen kann Folgendes definiert werden:

- Komplexitätsstufe und Gültigkeitsdauer des Passwortes
- Zugriffsrechte (Key-/Lockcode- oder Access-List-Konzept)
- Administrationsberechtigung
- Status des Benutzers
- Eigenschaften der zur Benutzerkennung gehörenden USER-Queue

B
- Startformat

B
- UTM-SAT-Administrationsberechtigung (BS2000/OSD)

B
- Benutzer-spezifische Sprachumgebung (BS2000/OSD)

B
- Art und Weise der Authentisierung (BS2000/OSD: Passwort, Magnetstreifenkarte, Kerberos-Principal)

B

Mindestens einer Benutzerkennung müssen Sie die Administrationsberechtigung erteilen, damit die Anwendung administriert werden kann. Sie können mehreren Kennungen die Administrationsberechtigung geben, damit mehrere Benutzer unter den jeweiligen Kennungen gleichzeitig Administrationsfunktionen aufrufen können. Analoges gilt unter BS2000/OSD für die UTM-SAT-Administrationsberechtigung und das Aufrufen von SAT-Preselection-Funktionen.

Eine Anwendung kann auch ohne Benutzerkennungen generiert werden. Der Benutzer muss sich dann nicht identifizieren und openUTM verwendet intern den Namen des jeweiligen Clients als Benutzerkennung. Damit kann jeder Benutzer Administrationskommandos und unter BS2000/OSD auch UTM-SAT-Administrationskommandos absetzen. Wird ohne Benutzerkennung gearbeitet, können einige Datenschutzfunktionen von openUTM nicht genutzt werden.

```
USER␣        username

             [ ,KSET=keysetname ]

             [ ,PASS={ ( password,DARK) |
                       ( *RANDOM,DARK ) |
                         password       |
                         *RANDOM } ]

             [ ,PERMIT={ NONE |ADMIN | SATADM[1] | ( ADMIN,SATADM)[1] }

             [ ,PROTECT-PW=( [ length ]
                           ,[ { NONE | MIN | MED | MAX } ]
                           ,[ maxtime ]
                            ,[ mintime ] ) ][2]

             [ ,QLEV=queue_level_number ]

             [ ,QMODE={ STD | WRAP-AROUND } ]

             [ ,Q-READ-ACL=read-keysetname ]

             [ ,Q-WRITE-ACL=write-keysetname ]

             [ ,RESTART={ YES | NO } ]

             [ ,STATUS={ ON | OFF } ]
```

B ***BS2000/OSD-spezifische Operanden***

```
B            [ ,CARD=( position,characterstring ) ]

B            [ ,CERTIFICATE={ *NONE |
B                             number [,CERTIFICATE-AUTHORITY = number1 ] } ]

B            [ ,FORMAT= { + | * | # }formatname ]

B            [ ,LOCALE=( [ lang_id ][,[ terr_id ][ ,ccsname ] ] ) ]

B            [ ,PRINCIPAL=characterstring ]

B            [ ,SATSEL={ NONE | BOTH | SUCC | FAIL } ]
```

B [1] Nur in BS2000/OSD erlaubt

[2] Kommas am Ende können entfallen, d.h. statt (8,NONE,,) können Sie (8,NONE) angeben.

username UTM-Benutzerkennung, die der Benutzer beim Anmelden an die Anwendung bzw. ein Client beim Aufbau einer Conversation mit der Anwendung angibt. *username* kann max. 8 Zeichen lang sein
(siehe auch LTERM-Anweisung, Operand USER=*username* ab Seite 354).

Der angegebene *username* muss eindeutig sein und darf keinem weiteren Objekt der Namensklasse 2 zugeordnet sein. Siehe dazu auch Abschnitt „Eindeutigkeit der Namen und Adressen“ auf Seite 262.

Ist *username* identisch mit dem Namen eines LTERM-Partners, der einer TS-Anwendung (PTYPE=APPLI oder SOCKET) oder einem remote-UPIC-Client (PTYPE=UPIC-R) zugeordnet ist, dann sind die bei LTERM gegebenen Hinweise auf Seite 365 zu berücksichtigen.

B CARD= (nur in BS2000/OSD erlaubt)
B vereinbart, dass beim Anmelden an die Anwendung mit dieser Benutzerkennung eine Prüfung einer Magnetstreifenkarte erfolgen soll und gibt an, auf welchen Inhalt die Ausweisinformation geprüft werden soll. Mit CARD geben Sie ein Teilfeld der auf der Magnetstreifenkarte abgespeicherten Information an, das von openUTM auf Übereinstimmung geprüft wird.

B Für den Parameter muss Folgendes gelten:
B pos + Länge(Zeichenfolge) -1 ≤ MAX CARDLTH
B Ansonsten wird der Parameter ignoriert.

B Die Angabe von CARD= schließt die Verwendung von CERTIFICATE= und
B PRINCIPAL= aus.

B position Beginn der zu prüfenden Information auf dem Ausweis:
B *position*=1 entspricht dem ersten Byte etc.

B characterstring
B openUTM prüft beim Anmelden an die Anwendung, ob die Ausweisinformation ab der angegebenen Position mit dieser Zeichenfolge übereinstimmt.

B *characterstring* kann wie folgt angegeben werden:

B – als hexadezimale Zeichenfolge; hexadezimale Zeichen müssen paarweise auftreten, z.B. X’DDEF’
B – als alphanumerische Zeichenfolge, z.B. FRIDOLIN oder C’@FRIEDEL’

B Sonderzeichen kann man nur in der Form C’...’ oder X’...’ eingeben.

B Standard: Keine Ausweisprüfung beim Anmelden an die Anwendung.
B Maximallänge: 100 Byte (siehe auch MAX ...,CARDLTH=)

CERTIFICATE= (nur in BS2000/OSD erlaubt)
gibt an, ob sich ein Benutzer beim Anmelden an die UTM-Anwendung mit einer Chip-Karte ausweisen muss, auf der das generierte Zertifikat enthalten ist. Das Zertifikat wird übergeben, wenn sich der Benutzer über einen Client ≠ Socket anmeldet, der über das Produkt TranSON mit der Anwendung verbunden ist. openUTM prüft beim Anmelden, ob das erhaltene Zertifikat mit dem generierten Zertifikat übereinstimmt. Weitere Informationen siehe auch openUTM-Handbuch „Konzepte und Funktionen“.

Die Angabe von CERTIFICATE= schließt die Verwendung von CARD=, PASS= und PRINCIPAL= aus.

*NONE Der Benutzer muss sich nicht über ein Zertifikat ausweisen.

Standard: *NONE

number Der Benutzer muss sich über ein Zertifikat ausweisen. Dabei ist *number* die Nummer des Zertifikats, über das sich der Benutzer ausweisen muss.

Minimum: 0
Maximum: 2147483647

CERTIFICATE-AUTHORITY=number1
Nummer der Zertifizierungsstelle, die das Zertifikat ausgestellt hat. Wird *number1* weggelassen oder wird *number1*=0 angegeben, dann wird die Nummer der Zertifizierungsstelle beim Anmelden nicht überprüft.

Standard: 0 (= keine Überprüfung)
Minimum: 0 bzw. 1 (für eine echte Nummer)
Maximum: 2147483647

FORMAT= Formatkennzeichen für ein Benutzer-spezifisches Startformat.
Dieses Startformat wird automatisch nach jedem erfolgreichen Anmelden an die Anwendung ausgegeben, wenn kein offener Vorgang für diesen Benutzer vorhanden ist. Befindet sich der Benutzer (USER) nach erfolgreicher Berechtigungsprüfung noch in einem Vorgang, so erscheint das Startformat **nicht**, stattdessen wird der letzte Dialog-Bildschirm ausgegeben (Vorgangswiederanlauf). Wird ein eigener Anmelde-Vorgang eingesetzt, kann der Name des Benutzer-spezifischen Startformats im zweiten Teil des Anmelde-Vorgangs mit einem SIGN ST-Aufruf abgefragt werden.

Das Formatkennzeichen setzt sich wie folgt zusammen:

+, * oder # gefolgt von einem max. 7 Zeichen langen alphanumerischen Namen (*formatname*).

#Formate können nur im Zusammenhang mit einem Anmelde-Vorgang genutzt werden.

Es bedeuten:

\+ Beim nächsten MGET-Aufruf des Teilprogramms werden im Nachrichtenbereich (NB) jedem Inhalt eines Formatfeldes 2 Byte für das Attributfeld vorangestellt, d.h. die Feldeigenschaften sind vom Teilprogramm veränderbar. Der Formatname an der KDCS-Schnittstelle ist +*formatname*.

* Dem Inhalt eines Formatfeldes wird im Nachrichtenbereich des nächsten MGET-Aufrufs des Teilprogramms kein Byte für ein Attributfeld vorangestellt, d.h. die Feldeigenschaften sind vom Teilprogramm nicht veränderbar. Der Formatname an der KDCS-Schnittstelle ist **formatname*.

\# bezeichnet ein Format mit erweiterten Benutzerattributen (**Extend**ed User Attributes). Feldeigenschaften und globale Formateigenschaften sind vom Teilprogramm veränderbar. Der Formatname an der KDCS-Schnittstelle ist #*formatname*.

Standard: Kein Startformat

KSET=keysetname

Name des Keysets, das der Benutzerkennung zugeordnet wird. Das Keyset wird mit der KSET-Anweisung definiert. Pro Benutzer (USER) kann max. ein Keyset zugeordnet werden.

Das Keyset legt die Zugriffsrechte dieses Benutzerkennung für die Nutzung von Services der Anwendung und von entfernten Services (LTAC) fest, die in dieser Anwendung generiert wurden.

Unter dieser Benutzerkennung können mit Lockcode bzw. Access-List geschützte Services der Anwendung nur gestartet bzw. mit Lockcode bzw. Access-List geschützte entfernte Services nur adressiert werden, wenn Folgendes zutrifft: Sowohl im zugeordneten Keyset der Benutzerkennung als auch im Keyset des LTERM-Partners, über den die Anmeldung unter der Benutzerkennung erfolgte, ist der zum Lockcode bzw. der Access-List passende Key- bzw. Zugangscode enthalten.

Das Lock-/Keycode- und das Access-List-Konzept werden ausführlich im openUTM-Handbuch „Konzepte und Funktionen“ beschrieben. Eine Einführung in die Zugriffskontrolle finden Sie im gleichnamigen Abschnitt ab Seite 216.

Services, deren Vorgangs-TACs nicht mit Codes gesichert sind, kann der Benutzer bzw. das Client-Programm ohne Einschränkung aufrufen.

Standard: kein Keyset, der Benutzer kann nur auf nicht mit Codes gesicherte Clients und LTERM-Partner zugreifen.

LOCALE=(lang_id,terr_id,ccsname)
definiert die Sprachumgebung des Benutzers.

lang_id Max. 2 Zeichen langes Sprachkennzeichen für die UTM-Benutzerkennung. Das Sprachkennzeichen ist frei wählbar.

Das Sprachkennzeichen kann von den Teilprogrammen der Anwendung abgefragt werden. So können die Teilprogramme Nachrichten an die Benutzerkennung in der Landessprache des Benutzers übertragen.

terr_id Max. 2 Zeichen langes Territorialkennzeichen für die UTM-Benutzerkennung. Das Territorialkennzeichen ist frei wählbar.

Das Territorialkennzeichen kann von den Teilprogrammen der Anwendung abgefragt werden. So können Sie in Nachrichten an den Benutzer die territorialen Besonderheiten in der Landessprache des Benutzers berücksichtigen.

ccsname (**c**oded **c**haracter **s**et **name**)
Für *ccsname* ist der max. 8 Zeichen lange Name eines erweiterten Zeichensatzes (CCS-Name) anzugeben. Der angegebene CCS-Name muss zu einem im BS2000-System definierten EBCDIC-Zeichensatz gehören (siehe auch Benutzerhandbuch „XHCS“).
Zum Zeitpunkt der Generierung kann openUTM die Gültigkeit des CCS-Namens im BS2000-System nicht überprüfen.

Der zum CCS-Namen gehörende Zeichensatz wird bei der Ausgabe von Dialog-Nachrichten verwendet, wenn der Benutzer an einem 8-Bit-Terminal angemeldet ist und kein anderer CCS-Name über Editprofil oder Format ausgewählt wird.
Der Zeichensatz muss kompatibel sein zu einem erweiterten ISO-Zeichensatz, der von diesem Terminal unterstützt wird. Zum Zeitpunkt der Generierung kann openUTM diese Kompatibilität nicht überprüfen, deshalb werden von KDCDEF auch falsche Angaben akzeptiert.

Standard:
Wird USER ...,LOCALE nicht angegeben, dann wird das in der MAX-Anweisung definierte Locale der Anwendung verwendet.

PASS= Max. 8 Zeichen langes Passwort, das der Benutzer bei der Berechtigungsprüfung angeben muss. Das angegebene Passwort muss der im Operanden PROTECT-PW= festgelegten Komplexitätsstufe genügen.

B B PASS= darf nicht zusammen mit CERTIFICATE= oder PRINCIPAL= verwendet werden.

Bei Angabe von *RANDOM wird für die Benutzerkennung ein zufälliges Passwort generiert, das niemandem bekannt ist. Der Benutzerkennung muss anschließend mit dem Tool KDCUPD oder per Administration ein gültiges Passwort übertragen werden. Ein durch *RANDOM erzeugtes Passwort unterliegt nicht den durch PROTECT-PW= gesetzten Bedingungen.

Es ist darauf zu achten, dass spätestens zum Startzeitpunkt zumindest eine Benutzerkennung mit Administrationsberechtigung konfiguriert ist, die kein per *RANDOM erzeugtes Passwort hat, da die Anwendung sonst nicht administrierbar ist.

Die Parameter haben folgende Bedeutung:

B *BS2000/OSD:*

B password
B *RANDOM
B Standardanmelde-Dialog:
B Der Benutzer muss zur Anmeldung 'KDCSIGN *username*,*password*' eingeben.

B Anmelde-Vorgang:
B Benutzerkennung und Passwort sind mit dem KDCS-Aufruf SIGN ON an
B openUTM zu übergeben.

B (password, DARK)
B (*RANDOM, DARK)
B Standardanmelde-Dialog:
B Der Benutzer gibt bei der Anmeldung zunächst nur 'KDCSIGN *username*'
B ein. openUTM fordert ihn dann auf, das Passwort in ein dunkelgesteuertes
B Feld des Bildschirms einzugeben.

B Anmelde-Vorgang:
B openUTM fordert den Benutzer im Zwischen-Dialog auf, das Passwort in ein
B dunkelgesteuertes Feld einzugeben.

B *password* kann wie folgt eingegeben werden:
B – hexadezimal, z.B. X'DDFF'
B – als Zeichenkonstante, z.B. C'@LKE'
B – als abdruckbare alphanumerische Zeichenfolge, z.B. NORBERT

B Standard: 8 Leerzeichen (d.h. kein Passwort)

X/W *Unix-, Windows-Systeme:*

X/W password
X/W *RANDOM
X/W (password,DARK)
X/W (*RANDOM,DARK)
X/W Standardanmelde-Dialog:
X/W Der Benutzer gibt bei der Anmeldung zunächst nur *username* an. openUTM
X/W fordert ihn dann auf, das Passwort einzugeben.

X/W Anmelde-Vorgang:
X/W openUTM fordert den Benutzer im Zwischen-Dialog auf, das Passwort wie
X/W beim Standardanmeldeverfahren anzugeben.

X/W Das Passwort kann als abdruckbare alphanumerische Zeichenfolge einge-
X/W ben werden, beispielsweise `UTM4EVER` oder `C'UTM␣EVER'`.

X/W Standard: 8 Leerzeichen (d.h. kein Passwort)

PERMIT= legt die Administrationsberechtigungsstufe des Benutzers innerhalb der lokalen Anwendung fest.

ADMIN Der Benutzer kann unter dieser Benutzerkennung Administrationsfunktionen ausführen.

NONE Der Benutzer darf keine Administrationsfunktionen ausführen.

Standard: NONE

B Unter BS2000/OSD darf der Benutzer darüber hinaus keine SAT-Preselec-
B tion-Funktionen ausführen.

B SATADM Der Benutzer kann SAT-Preselection-Funktionen (UTM-SAT-Administration)
B ausführen.

B (ADMIN,SATADM)
B Der Benutzer kann Administrations- und SAT-Preselection-Funktionen aus-
B führen.

B PRINCIPAL=characterstring
B (nur im BS2000/OSD erlaubt)
B Die Authentisierung des Benutzers soll über Kerberos erfolgen. Die Au-
B thentisierung über Kerberos ist nur möglich, wenn die Anmeldung des Be-
B nutzers direkt von einem Kerberos-fähigen Terminal aus erfolgt (nicht über
B OMNIS).
B openUTM speichert die Kerberos-Information in der Länge ab, die sich aus
B dem Maximum der bei MAX PRINCIPAL-LTH und MAX CARDLTH gene-
B rierten Länge ergibt. Wenn die Kerberos-Information länger ist, wird sie auf
B diese Länge verkürzt abgespeichert.

Mit dem KDCS-Aufruf INFO (KCOM=CD) kann ein Teilprogrammlauf diese Information lesen, so lange der Benutzer an diesem Client angemeldet ist.

Die Angabe von PRINCIPAL schließt die Verwendung der Parameter CERTIFICATE, CARD und PASS aus.

characterstring muss wie folgt als alphanumerische Zeichenfolge, in Hochkommata eingeschlossen, angegeben werden:

```
C'windowsaccount@NT-DNS-REALM-NAME'
```

windowsaccount
: Domänen-Kennung des Benutzers

NT-DNS-REALM-NAME
: DNS-Name der Active-Directory-Domäne. Dieser Name ist ein fester Wert für jede Active-Directory-Domäne, der schon beim Einrichten des Kerberos-Schlüssels vergeben wurde.

  Die Länge des angegebenen Character-Strings darf nicht größer sein, als der bei MAX PRINCIPAL-LTH angegebene Wert. Andernfalls wird der Parameter ignoriert.

  openUTM speichert die Kerberos-Information in der Länge ab, die sich aus dem Maximum der bei MAX PRINCIPAL-LTH und MAX CARDLTH generierten Länge ergibt. Wenn die Kerberos-Information länger ist, wird sie auf diese Länge verkürzt abgespeichert.

  Mit dem KDCS-Aufruf INFO (KCOM=CD) kann ein Teilprogrammlauf diese Information lesen, so lange der Benutzer an diesem Client angemeldet ist.

Standard: keine Kerberos-Authentisierung
Maximallänge: der mit MAX ...,PRINCIPAL-LTH generierte Wert, siehe Seite 392

PROTECT-PW=
: legt die minimale Länge, die Komplexitätsstufe sowie Mindest- und Höchstgültigkeitsdauer des Benutzer-Passwortes fest. Die für PROTECT-PW angegebenen Werte müssen Sie bei der Angabe des Passworts im Operanden PASS= berücksichtigen. openUTM prüft diese auch bei der Änderung eines Passworts durch den Administrator (Administrationskommando KDCUSER, siehe openUTM-Handbuch „Anwendungen administrieren“) oder durch ein Teilprogramm (SIGN CP-Aufruf).

  length
  : gibt die Länge in Anzahl Zeichen an, die das Passwort mindestens haben muss. Der Administrator kann das Passwort eines Benutzers nur löschen, wenn für *length*=0 angegeben wird.

    Standard: 0

Minimalwert:
0 bei NONE oder keine Angabe für die Komplexitätsstufe
1 bei MIN
2 bei MED
3 bei MAX

Maximalwert: 8

NONE/MIN/MED/MAX
bezeichnen die Komplexitätsstufe des Passworts.

NONE Es kann jede beliebige Zeichenfolge als Passwort angegeben werden.

Standard: NONE

MIN In dem Passwort dürfen maximal 2 aufeinanderfolgende Zeichen gleich sein. Die minimale Länge des Passworts ist ein Zeichen.

MED In dem Passwort dürfen maximal 2 aufeinanderfolgende Zeichen gleich sein. Das Passwort muss mindestens einen Buchstaben und eine Ziffer enthalten. Die minimale Länge des Passworts ist 2 Zeichen.

MAX In dem Passwort dürfen maximal 2 aufeinanderfolgende Zeichen gleich sein. Das Passwort muss mindestens einen Buchstaben, eine Ziffer und ein Sonderzeichen enthalten. Die minimale Länge des Passworts ist 3 Zeichen. Sonderzeichen sind alle Zeichen, die von a-z, A-Z, 0-9 und Leerzeichen verschieden sind.

maxtime Maximale Gültigkeitsdauer:
maxtime gibt an, wieviele Tage das Passwort maximal gültig ist.

Wird *maxtime* angegeben, so läuft die Gültigkeit des Passworts am Ende des letzten Tages der Gültigkeitsdauer ab. Wird z.B. eine Gültigkeit von einem Tag generiert, so läuft die Gültigkeit um 24:00 Uhr des folgenden Tages ab.

Wurde die Anwendung mit SIGNON GRACE=YES generiert, so wird bei einer Neu-Generierung das Passwort auf „abgelaufen“ gesetzt; der Benutzer muss dann beim ersten Anmelden ein neues Passwort vergeben.

Ist die Gültigkeit abgelaufen, hängt es davon ab, wie die UTM-Anwendung generiert ist:

– mit Grace-Signon (SIGNON GRACE=YES)
  Der Benutzer kann und muss das Passwort bei der nächsten Anmeldung selbst ändern, sofern ihm das Anmeldeverfahren die Möglichkeit zum Ändern des Passworts gibt. Ist das nicht der Fall, dann muss das Passwort administrativ geändert werden, da sonst kein Anmelden unter

dieser Benutzerkennung mehr möglich ist. Dieser Fall kann z.B. eintreten bei Benutzern, die sich über TS-Anwendungen und UPIC-Clients ohne Anmelde-Vorgang oder über einen OSI-TP-Partner anmelden.

– ohne Grace-Signon (SIGNON GRACE=NO)
openUTM lehnt eine Anmeldung mit der Meldung K120 ab. Der Administrator muss dann das Passwort ändern.

Wird *maxtime*=0 angegeben, so ist die Gültigkeitsdauer des Passworts nicht beschränkt.

Standard: 0 (Gültigkeitsdauer wird nicht beschränkt)
Maximalwert: 180
Minimalwert: 0

mintime minimale Gültigkeitsdauer:
Mit *mintime* legen Sie die minimale Gültigkeitsdauer des Passworts in Tagen fest. Nach der Änderung des Passworts darf der Benutzer das Passwort frühestens nach Ablauf der minimalen Gültigkeitsdauer erneut ändern.

Durch die Angabe von *mintime* > 0 können Sie verhindern, dass ein Benutzer, dessen Passwort abgelaufen ist, zweimal hintereinander sein Passwort ändert, und damit wieder sein ursprüngliches (= abgelaufenes) Passwort einstellt.

Wird eine minimale Gültigkeitsdauer von einem Tag angegeben, so darf das Passwort frühestens um 0.00 Uhr des folgenden Tages geändert werden (Ortszeit der Generierung).

Nach einer Passwortänderung durch den Administrator bzw. nach einer Neugenerierung kann der Benutzer das Passwort immer ändern, unabhängig davon, ob die minimale Gültigkeitsdauer abgelaufen ist oder nicht.

mintime darf nicht größer als *maxtime* (maximale Gültigkeitsdauer) sein.
Wird *mintime*=0 angegeben, so ist die minimale Gültigkeitsdauer des Passworts nicht beschränkt.

Standard: 0 (keine Beschränkung)
Minimalwert: 0
Maximalwert: 180

QLEV=queue_level_number
(**q**ueue **lev**el)
Gibt an, wieviele asynchrone Nachrichten in der Message Queue des Benutzers (= USER-Queue) maximal zwischengespeichert werden können. Mit QLEV kann eine zu starke Belastung des Pagepool durch Nachrichten für diesen USER verhindert werden.

openUTM berücksichtigt die Asynchron-Aufträge erst am Ende der Transaktion. Daher kann die in QLEV festgelegte Anzahl von Nachrichten für eine Message Queue überschritten werden, wenn in einer Transaktion mehrere Nachrichten für dieselbe Queue erzeugt wurden.
Wird der Schwellwert überschritten, dann hängt das Verhalten vom Wert des Operanden QMODE= ab, siehe unten.

Bei QLEV=0 können keine Nachrichten in der Queue gespeichert werden; bei QLEV=32767 ist die Queue-Länge nicht begrenzt.

Standard: 32767
Minimalwert: 0
Maximalwert: 32767

Wird ein Wert angegeben, der größer als der Maximalwert ist, dann wird im KDCDEF-Lauf ohne Meldung der Standardwert gesetzt.

QMODE = (**Q**ueue **Mode**)
Bestimmt das Verhalten von openUTM für den Fall, dass bereits die maximal erlaubte Anzahl von Nachrichten in der USER-Queue gespeichert und somit der Queue-Level (Operand QLEV=) erreicht ist.

STD wenn der Queue-Level erreicht ist, lehnt openUTM weitere Nachrichten für die Queue mit einem negativen Returncode ab (40Z bei DPUT).

WRAP-AROUND
openUTM nimmt auch dann noch Nachrichten für die Queue an, wenn der Queue Level bereits erreicht ist. Beim Schreiben einer Nachricht in die Queue löscht openUTM dann die älteste der in der Queue stehenden Nachrichten.

Standard: STD

Q-READ-ACL=read-keysetname
Legt Lese- und Lösch-Rechte in der USER-Queue für fremde Benutzer fest. Für *read-keysetname* ist ein Keyset anzugeben, das mit einer KSET-Anweisung generiert worden ist.

Geben Sie Q-READ-ACL= an, dann kann ein fremder Benutzer (≠*username*) nur dann lesend auf die Queue zugreifen, wenn sowohl das Keyset seiner Benutzerkennung als auch das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, jeweils mindestens einen Keycode des Keysets *read-keysetname* enthält.

Der Eigentümer (*username*) der USER-Queue hat immer Lese- und Löschrechte, auch wenn die Rechte mit Q-READ-ACL eingeschränkt werden.

Geben Sie Q-READ-ACL= nicht an, dann hat jeder Benutzer Lese- und Lösch-Rechte in der Queue.

Standard: kein Keyset

Q-WRITE-ACL=write-keysetname

Legt Schreibrechte in der USER-Queue für fremde Benutzer fest.
Für *write-keysetname* ist ein Keyset anzugeben, das mit einer KSET-Anweisung generiert worden ist.

Geben Sie Q-WRITE-ACL= an, dann kann ein fremder Benutzer (≠*username*) nur dann schreibend auf die Queue zugreifen, wenn sowohl das Keyset seiner Benutzerkennung als auch das Keyset des LTERM-Partners, über den der Benutzer angemeldet ist, jeweils mindestens einen Keycode des Keysets *write-keysetname* enthält.

Der Eigentümer (*username*) der USER-Queue hat immer Schreibrechte, auch wenn die Rechte mit Q-WRITE-ACL eingeschränkt wurden.

Geben Sie Q-WRITE-ACL= nicht an, dann hat jeder Benutzer Schreibrechte in der Queue.

Standard: kein Keyset

RESTART= gibt an, ob openUTM für diese Benutzerkennung einen automatischen Vorgangswiederanlauf durchführt.

YES Der zu dieser Benutzerkennung gehörige Vorgangskontext wird gesichert. Damit kann für Benutzer, die sich unter dieser Benutzerkennung anmelden, ein Vorgangswiederanlauf durchgeführt werden, wenn ein offener Vorgang für diese Benutzerkennung existiert.

RESTART=YES ist in UTM-Cluster-Anwendungen nur erlaubt, wenn Sie in der CLUSTER-Anweisung USER-RESTART=YES angegeben haben.

Beim Vorgangswiederanlauf spielen noch der Typ des Client und eventuell generierte Anmelde-Vorgänge eine Rolle. Nähere Informationen finden Sie in Abschnitt „Wiederanlauf generieren“ auf Seite 147 und im openUTM-Handbuch „Einsatz von openUTM-Anwendungen“.

Standard in stand-alone Anwendungen.

NO Der zu dieser Benutzerkennung gehörige Vorgangskontext wird nicht gesichert, es ist kein Vorgangswiederanlauf möglich, d.h.

– Wird die Verbindung im laufenden Betrieb durch KDCOFF bzw. Verbindungsverlust abgebaut oder die Anwendung normal beendet, so wird der Vorgang auf den letzten Sicherungspunkt zurückgesetzt, beendet und danach der Event-Exit VORGANG mit KCKNZVG=D (=Disconnect) aufgerufen.

– Nach der abnormalen Beendigung einer Anwendung wird beim UTM-Warmstart ein noch offener Vorgang für diesen UTM-Benutzer beendet, ohne dass der Event-Exit VORGANG aufgerufen wird.

– KDCDISP/KDCLAST zeigt nach Verbindungsaufbau das gleiche Verhalten wie nach einer Neugenerierung.

Wird RESTART=NO zusammen mit SIGNON MULTI-SIGNON=YES angegeben, dann können sich mehrere Benutzer gleichzeitig unter dieser Benutzerkennung bei openUTM anmelden, wobei sich nur einer der Benutzer am Terminal anmelden darf. Es können sich jedoch beliebig viele Client-Programme gleichzeitig anmelden.

Standard in UTM-Cluster-Anwendungen.

B SATSEL= steuert die Art der SAT-Protokollierung der Ereignisse für diesen Benutzer.

B Bei eingeschalteter SAT-Protokollierung (MAX ...,SAT=YES) werden dann alle Ereignisse, die dieser Benutzer auslöst, entsprechend den Angaben für SATSEL protokolliert.

B Die Art der SAT-Protokollierung kann allgemein für alle TACs und Benutzer in der SATSEL-Anweisung festgelegt werden. Mit USER ...,SATSEL= wird zusätzlich zu den Angaben in der SATSEL-Anweisung eine Benutzer-spezifische Protokollierung festgelegt. Wird die Protokollierung einer Ereignisklasse in der SATSEL-Anweisung ausgeschlossen, so werden Ereignisse dieser Klasse nicht protokolliert (zur Verknüpfung der EVENT-, TAC- und USER-spezifischen Einstellung siehe openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“).

B SATSEL kann auch bei ausgeschalteter SAT-Protokollierung (MAX ...,SAT=OFF) generiert werden. Die Anweisungen werden dann beim Start der Anwendung nicht wirksam. Sie erzeugen so eine Vorbelegung für eine SAT-Protokollierung, die im laufenden Betrieb bei Bedarf eingeschaltet werden kann (UTM-SAT-Administrationskommando KDCMSAT, siehe openUTM-Handbuch „Einsatz von openUTM-Anwendungen unter BS2000/OSD“).

B NONE Es wird keine Benutzer-spezifische Art der SAT-Protokollierung definiert.

B Standard: NONE

B BOTH Es werden erfolgreiche **und** nicht erfolgreiche Ereignisse protokolliert.

B SUCC Es werden nur erfolgreiche Ereignisse protokolliert.

B FAIL Es werden nur nicht erfolgreiche Ereignisse protokolliert.

STATUS= gibt an, ob die Benutzerkennung beim Start der Anwendung zugelassen oder gesperrt ist.

ON Die Benutzerkennung ist zugelassen.

Standard: ON

OFF Die Benutzerkennung ist gesperrt. Es kann sich kein Benutzer oder Client mit dieser Benutzerkennung anmelden, bis er vom Administrator zugelassen wird.

Benutzerkennungen, die implizit oder explizit über LTERM-Anweisungen (LTERM ...,USER=) einem UPIC-Client oder einer TS-Anwendung zugeordnet sind, sind immer gesperrt. Sie können auch nicht vom UTM-Administrator zugelassen werden. Diese Benutzerkennungen werden Verbindungs-Benutzerkennungen genannt.

UTMD - Anwendungsparameter für verteilte Verarbeitung festlegen

Mit der UTMD-Anweisung werden für die verteilte Verarbeitung global gültige Werte der lokalen UTM-Anwendung festgelegt. Eine UTMD-Anweisung ist nur erforderlich für Anwendungen, die das LU6.1- oder das OSI TP-Protokoll zur Kommunikation verwenden.

Die UTMD-Anweisung darf nur einmal angegeben werden.

Wenn Sie in Ihrer Anwendung das OSI TP-Protokoll nutzen, dann können Sie in der UTMD-Anweisung den Application Process Title (APT) der Anwendung angeben. Der APT wird von einigen heterogenen Partnern benötigt, die eine andere Ausprägung des OSI TP-Protokolls unterstützen. Diese Anwendungen erwarten die Angabe des Application Process Title beim Verbindungsaufbau.

Der APT bildet zusammen mit jedem Application Entity Qualifier (AEQ), den Sie einem Zugriffspunkt (Access Point) Ihrer Anwendung zugeordnet haben, einen im OSI-Netz eindeutigen Application Entity Title (AET). Mit ihm kann die Partner-Anwendung den Zugriffspunkt der lokalen Anwendung identifizieren, über den die Kommunikation erfolgen soll.

```
UTMD␣          [ APPLICATION-PROCESS-TITLE=object_identifier ]
               [ ,CONCTIME={ time1 | ( time1,time2 ) } ]
               [ ,MAXJR=%_maxjr ]
               [ ,PTCTIME=time3 ]
               [ ,RSET={ GLOBAL | LOCAL } ]
```

APPLICATION-PROCESS-TITLE=object_identifier

(nur relevant, wenn in der Anwendung das OSI TP-Protokoll benutzt wird) Adresskomponente des Application Entity Title (AET). Der AET wird dann benötigt, wenn mit Transaktionssicherung (Commit Functional Unit) gearbeitet wird, oder aber ein heterogener Partner einen AET für den Verbindungsaufbau erwartet.

object_identifier ist der Application Process Title (APT) Ihrer Anwendung. Wenn er nicht von einem Standardisierungsgremium vergeben wurde, müssen Sie bei der Vergabe die bereits bestehenden Konventionen für *komponente1* und *komponente2* beachten. Siehe dazu Abschnitt „Application Entity Title (AET)“ auf Seite 91. In der Praxis ist es erforderlich, dass der angegebene *object_identifier* im Netz eindeutig ist.

Enthält der mit einer Partner-Anwendung vereinbarte Application Context (Definition des Kommunikationspartners in der OSI-LPAP-Anweisung, Seite 429) die CCR-Syntax, dann muss ein APT angegeben werden.

Ein Application Process Title besteht aus mindestens 2, maximal aber aus 10 Komponenten. Er wird angegeben in der Form:

(*komponente1,komponente2,...,komponente10*)

Für die Komponenten werden positive ganze Zahlen vergeben. Bestimmten Zahlen einzelner Komponenten sind symbolische Namen zugeordnet. Sie können statt der Zahlen auch diese Namen angeben. Bei einem Application Process Title ist sowohl die Position der einzelnen Komponenten in der Klammer als auch die Anzahl der Komponenten relevant, z.B. bezeichnen (1,2,3), (1,2,3,0,0) und (0,1,2,3,0) verschiedene Application Process Titles.

openUTM und die OSI-Norm erlauben für *komponente1* nur die Werte bzw. die symbolischen Namen:
0 oder CCITT
1 oder ISO
2 oder JOINT-ISO-CCITT

Die für *komponente2* erlaubten Werte sind abhängig vom Wert der Komponente *komponente1*.

- Ist *komponente1*=0 oder 1, dann sind für *komponente2* Werte zwischen 0 und 39 erlaubt (0 ≤ *komponente2* ≤ 39).
- Ist *komponente1*=2, dann sind für *komponente2* Werte zwischen 0 und 67108863 (2^{26}-1) erlaubt (0 ≤ *komponente2* ≤ 67108863).

Für alle anderen Komponenten sind Werte zwischen 0 und 67108863 (2^{26}-1) erlaubt.

openUTM führt keine Überprüfung durch, ob ein von einem Standardisierungsgremium registrierter Application Process Title angegeben wird.

CONCTIME= (**con**nection **c**ontrol **time**)

time1 Zeit in Sekunden zur Überwachung des Aufbaus einer Session (LU6.1) bzw. Association (OSI TP). Wenn die Session bzw. Association nicht innerhalb der angegebenen Zeit aufgebaut wird, baut openUTM die Transportverbindung ab. Damit wird verhindert, dass eine Transportverbindung wegen eines misslungenen Aufbaus einer Session bzw. Association blockiert bleibt. Das ist möglich, wenn eine zum Aufbau erforderliche Nachricht verloren geht.

Bei CONCTIME=0 für LU6.1 wird der Aufbau nicht überwacht.
Bei CONCTIME=0 für OSI TP wird die Zeitüberwachung intern auf 60 Sek. gesetzt.

Standard: 0
Minimalwert: 0
Maximalwert: 32767

time2 Zeit in Sekunden, die beim Übertragen einer asynchronen Nachricht maximal auf eine Quittung von der Partner-Anwendung gewartet wird. Nach Ablauf der angegebenen Zeit baut openUTM die Transportverbindung ab. Der Auftrag geht nicht verloren. Mit der Überwachung wird vermieden, dass eine Verbindung nicht weiter genutzt werden kann, weil eine Quittung verloren ging, oder ein Verbindungsverlust nicht vom Transportsystem an openUTM gemeldet wurde. Der Wert 0 bedeutet, dass keine Überwachung durchgeführt wird.

Standard: 0
Minimalwert: 0
Maximalwert: 32767

MAXJR=%_maxjr

(**max**imal number of **j**ob **r**eceivers)
legt fest, wieviele Auftragnehmer-Vorgänge maximal zu einem Zeitpunkt in der lokalen Anwendung addressiert sein dürfen.
Diese Zahl entspricht den gleichzeitig aktiven APRO-Aufrufen.

Der %-Wert bezieht sich auf die Anzahl der generierten Sessions und Associations (Anzahl LSES-Anweisungen für das Protokoll LU6.1 + Summe der bei den OSI-LPAP-Anweisungen angegebenen Anzahl paralleler Verbindungen; Operand ASSOCIATIONS). Der %-Wert muss zwischen 0 und 200 liegen. Ein Wert > 100 wird angegeben, damit APRO-Aufrufe, die vor der Sessionbelegung abgesetzt werden, in eine Tabelle eingetragen werden können.

Standard: 100
d.h. die maximale Anzahl der zu einer Zeit aktiven Auftragnehmer-Vorgänge ist gleich der Anzahl der Sessions und Associations.

Minimalwert: 0
Maximalwert: 200

PTCTIME=time3

(**p**repare **t**o **c**ommit)
legt die Zeit in Sekunden fest, die ein Server-Vorgang maximal im Zustand PTC (Transaktionsstatus P) auf eine Quittung vom Auftraggeber wartet. Nach Ablauf der Zeit wird die Verbindung zum Auftraggeber abgebaut, die Transaktion im Server-Vorgang zurückgesetzt und der Vorgang beendet. Dadurch kann es evtl. zu einem Mismatch kommen. Bei PTCTIME=0 wird beliebig lange auf eine Quittung gewartet.

Standard:
Wert bei MAX ...,TERMWAIT=*time* für die Wartezeit nach PEND KP.

Minimalwert: 0
Maximalwert: 32767

RSET= legt bei der verteilten Transaktionsverarbeitung fest, wie sich das Rücksetzen einer lokalen Transaktion auf die verteilte Transaktion auswirkt.

Eine lokale Transaktion kann zurückgesetzt werden:
- durch einen RSET-Aufruf aus einem Teilprogramm oder
- durch das Rücksetzen einer Datenbanktransaktion, die an der lokalen Transaktion beteiligt ist.

GLOBAL Nach dem Rücksetzen einer lokalen Transaktion muss das Teilprogramm so beendet werden, dass openUTM die verteilte Transaktion zurücksetzt.

Standard: GLOBAL

LOCAL Das Rücksetzen einer lokalen Transaktion hat keinen Einfluss auf die verteilte Transaktion.

Es kann zu Inkonsistenzen in den verteilten Datenbeständen kommen, wenn einige der an einer verteilten Transaktion beteiligten lokalen Transaktionen zurückgesetzt und andere abgeschlossen werden. Wird RSET=LOCAL angegeben, wird die globale Datenkonsistenz nicht mehr von den beteiligten Systemkomponenten garantiert. Sie liegt dann in der Verantwortung der Anwendungsteilprogramme. Sie müssen entscheiden, in welchen Situationen die verteilte Transaktion noch sinnvoll beendet werden kann und in welchen Situationen sie zurückgesetzt werden muss.

6.6 Dialog-Führung - Einfluss von Generierungsparametern

Auf die Dialog-Führung kann bei der Generierung durch folgende Anweisungen und Parameter des Generierungstools KDCDEF Einfluss genommen werden:

KDCDEF-Anweisung	**Auswirkung**
EXIT ...,USAGE=INPUT	Event-Exit INPUT
LTAC ..., WAITTIME=	maximale Wartezeiten für die verteilte Verarbeitung
LTERM ..., RESTART=	betrifft asynchrone Nachrichten und Vorgangswiederanlauf, wenn keine Benutzerkennungen generiert sind
LTERM ..., USER=	Automatisches KDCSIGN
MAX ..., APPLIMODE=	betrifft Vorgangswiederanlauf und asynchrone Nachrichten
MAX ..., CONN-USERS=	Auslastung der Anwendung steuern: Anzahl der gleichzeitig aktiven Benutzer oder Clients beschränken
MAX ..., NRCONV=	maximale Zahl gekellerter Vorgänge
MAX ..., PGWTIME=	Zeit in Sekunden, die ein Teilprogramm nach einem blockierenden Aufruf maximal auf das Eintreffen von Nachrichten warten darf.
MAX ..., TERMWAIT=	maximale Wartezeit bis zur nächsten Eingabe vom Terminal innerhalb einer Mehrschritt-Transaktion (nach PEND KP)
PTERM ..., IDLETIME= TPOOL ..., IDLETIME=	maximale Wartezeit bis zur nächsten Eingabe vom Client nach Transaktionsende bzw. nach der Anmeldung (nach PEND RE/FI/ER)
SIGNON ..., GRACE=	der Benutzer kann sein Passwort nach Ablauf der Gültigkeit noch ändern oder nicht
SIGNON ..., MULTI-SIGNON=	es können sich mehrere Benutzer/Clients gleichzeitig mit derselben Benutzerkennung anmelden oder nicht
SIGNON ..., SILENT-ALARM=	Beschränkung ungültiger Anmeldeversuche
USER-Anweisungen vorhanden	openUTM führt für alle Benutzer die Berechtigungsprüfung durch
USER ..., PASS=	Berechtigungsprüfung mit Passwort, Eingabe evtl. dunkel gesteuert
USER ..., PROTECT-PW=	Prüfung der Zugangsberechtigung mit dunkelgesteuertem Passwort, Gültigkeitsdauer und Komplexitätsstufe des Passwortes
USER ..., RESTART=	betrifft Vorgangswiederanlauf
SFUNC ...	F-Taste und (in BS2000/OSD) K-Taste wirkt als TAC, UTM-Kommando oder Kellerungswunsch
TAC KDCBADTC	Event-Service BADTACS wirksam
TAC KDCSGNTC	Anmeldung mit Anmelde-Vorgang

	KDCDEF-Anweisung	Auswirkung
B	***In BS2000/OSD sind darüber hinaus folgende Parameter von Bedeutung***	
B B	LTERM ..., ANNOAMSG=	asynchrone Nachrichten werden mit oder ohne Vorankündigung ausgegeben
B	LTERM ..., FORMAT=	LTERM-spezifisches Startformat
B B	TPOOL ..., ANNOAMSG=	Asynchrone Nachrichten werden vor der Ausgabe in der Systemzeile am Terminal ankündigt
B B	PTERM ..., CONNECT=Y	openUTM baut Verbindung zum Terminal beim Start der UTM-Anwendung automatisch auf
B	MAX ..., LOCALE=	Standard-Sprachumgebung der Anwendung festlegen
B B	LTERM ..., LOCALE=	Sprachumgebung der Clients festlegen, die sich über LTERM-Partner anschließen
B	TPOOL ..., FORMAT=	definiert das Startformat für Benutzer am Terminal
B B	TPOOL ..., LOCALE=	Sprachumgebung der Clients festlegen, die sich über LTERM-Pool anschließen
B	USER ..., CARD=	Berechtigungsprüfung mit Einlegen einer Magnetstreifenkarte
B B B	USER ..., CERTIFICATE=	gibt an, ob sich ein Benutzer beim Anmelden an die UTM-Anwendung mit einer Chip-Karte ausweisen muss, auf der das generierte Zertifikat enthalten ist.
B	USER ..., FORMAT=	User-spezifisches Startformat
B	USER ..., LOCALE=	Sprachumgebung des Benutzers festlegen
B	USER ..., PRINCIPAL=	die Authentisierung des Benutzers soll über Kerberos erfolgen.

6.7 Beispielgenerierung *Comfo*TRAVEL

Für die Beispielgenerierung *Comfo*TRAVEL wurde ein Reise-Buchungssystem konzipiert, das es Reisenden ermöglichen soll, über Reisebüros in München, Paris und New York, Reisen „aus einem Guss“ mit Hotel, Flug und Freizeitgestaltung, inklusive der nötigen Bankabwicklung zu buchen. Die Reisebüros greifen dabei auf ein zentrales Buchungssystem (RMS Reservation Management System) zu.

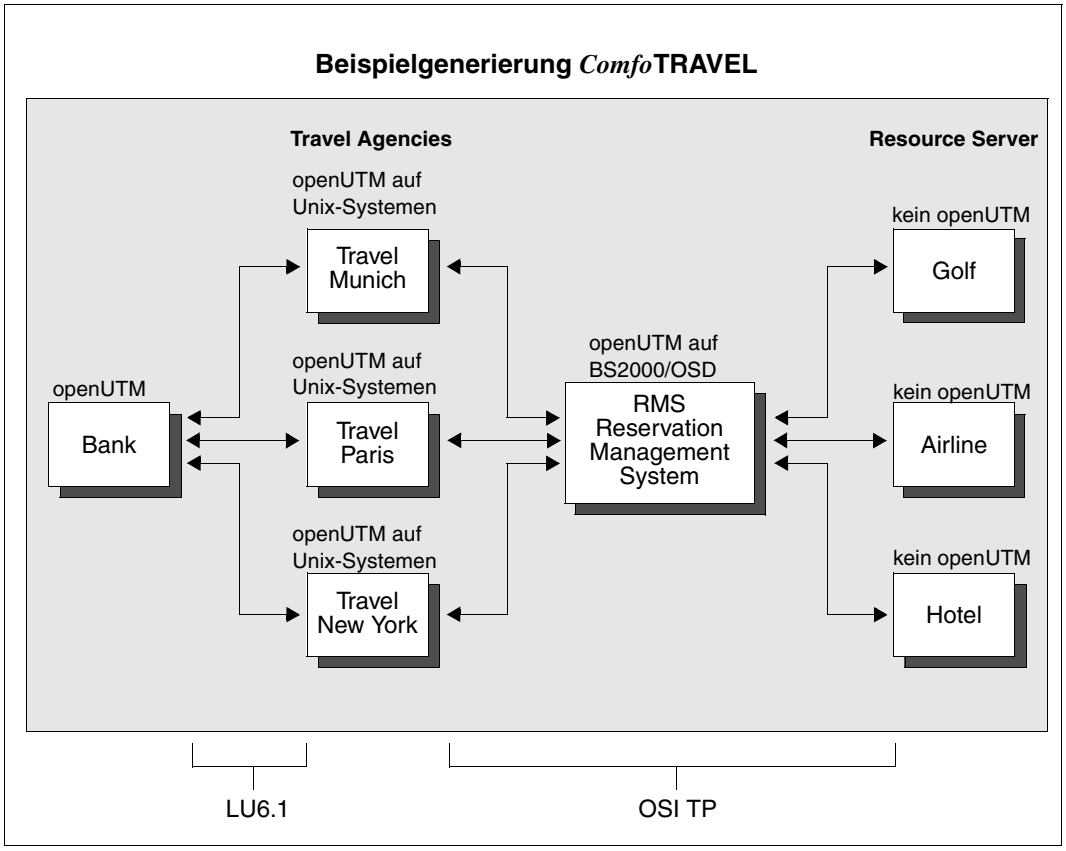


Bild 19: Beispielgenerierung *Comfo*TRAVEL mit verwendeten Protokollen

6.7.1 KDCDEF-Inputdatei DYNAMIC.RMS für UTM-D-Anwendung RMS

```
************************************
*       ADMINISTRATION Programs     *
************************************

PROGRAM KDCADM, COMP=ILCS
PROGRAM KDCDADM,COMP=ILCS
PROGRAM KDCPADM,COMP=ILCS

************************************
*       RMS Programs                *
************************************

PROGRAM AVALRESP, COMP=ILCS, LOAD-MODULE=RMS
PROGRAM RESRRESP, COMP=ILCS, LOAD-MODULE=RMS
PROGRAM CNCLRESP, COMP=ILCS, LOAD-MODULE=RMS
PROGRAM AUTHRESP, COMP=ILCS, LOAD-MODULE=RMS
PROGRAM INITRESP, COMP=ILCS, LOAD-MODULE=RMS
PROGRAM SHUTRESP, COMP=ILCS, LOAD-MODULE=RMS
PROGRAM ENQRRESP, COMP=ILCS, LOAD-MODULE=RMS
PROGRAM HNDLEXIT, COMP=ILCS, LOAD-MODULE=RMS

************************************
*       BOOKKEEPING Programs        *
************************************

PROGRAM BOOKKEEP, COMP=ILCS, LOAD-MODULE=BOOKKEEP

************************************
*       PERSONNEL Programs          *
************************************

PROGRAM PERSNNL, COMP=ILCS, LOAD-MODULE=PERSNNL

************************************
*       OFFICE Programs             *
************************************

PROGRAM OFFICE,  COMP=ILCS, LOAD-MODULE=OFFICE

********************************************
*** TAC-Statements ***
********************************************

******* TAC  BOOKKEEP      *****
*
TAC BOOKKEEP, PROGRAM=BOOKKEEP, LOCK=7
```

```
******* TACS PERSNNL       *****
*
TAC OPERSNNL, PROGRAM=PERSNNL, LOCK=2
TAC MPERSNNL, PROGRAM=PERSNNL, LOCK=3
TAC CPERSNNL, PROGRAM=PERSNNL, LOCK=4

******* TACS OFFICE        *****
*
TAC OFFCHRG , PROGRAM=OFFICE , LOCK=5
TAC OFFADMIN, PROGRAM=OFFICE , LOCK=6
*
********* TACS RMS           *****
*
TAC AVALRESP, PROGRAM=AVALRESP, LOCK=8
TAC RESRRESP, PROGRAM=RESRRESP, LOCK=8
TAC CNCLRESP, PROGRAM=CNCLRESP, LOCK=8
TAC AUTHRESP, PROGRAM=AUTHRESP, LOCK=8
TAC INITRESP, PROGRAM=INITRESP, LOCK=8
TAC SHUTRESP, PROGRAM=SHUTRESP, LOCK=8
TAC ENQRRESP, PROGRAM=ENQRRESP, LOCK=8

*** ADMINISTRATION DIALOG ***

DEFAULT TAC ADMIN=Y, PROGRAM=KDCADM

TAC KDCTAC  , LOCK=1
TAC KDCLTERM, LOCK=1
TAC KDCPTERM, LOCK=1
TAC KDCSWTCH, LOCK=1
TAC KDCSEND , LOCK=1
TAC KDCAPPL , LOCK=1
TAC KDCUSER , LOCK=1
TAC KDCDIAG , LOCK=1
TAC KDCLOG  , LOCK=1
TAC KDCINF  , LOCK=1
TAC KDCHELP , LOCK=1
TAC KDCLPAP , LOCK=1
TAC KDCLTAC , LOCK=1
TAC KDCSHUT , LOCK=1
TAC KDCTCL  , LOCK=1

TAC TACDADM  , PROGRAM=KDCDADM, LOCK=1
TAC TACPADM  , PROGRAM=KDCPADM, LOCK=1
```

```
*** ADMINISTRATION ASYNCHRON ***

DEFAULT TAC TYPE=A

TAC KDCTACA , LOCK=1
TAC KDCLTRMA, LOCK=1
TAC KDCPTRMA, LOCK=1
TAC KDCSWCHA, LOCK=1
TAC KDCUSERA, LOCK=1
TAC KDCSENDA, LOCK=1
TAC KDCAPPLA, LOCK=1
TAC KDCDIAGA, LOCK=1
TAC KDCLOGA , LOCK=1
TAC KDCINFA , LOCK=1
TAC KDCHELPA, LOCK=1
TAC KDCLPAPA, LOCK=1
TAC KDCLTACA, LOCK=1
TAC KDCSHUTA, LOCK=1
TAC KDCTCLA , LOCK=1
*
*
*
*
*

*********************************************
*** USER-Statements                       ***
*********************************************

USER SUPERUSR, PERMIT=ADMIN,    PASS='$23ADM--', PROTECT-PW=(8, MAX) -
             , KSET=MASTER
USER UTMADMIN, PERMIT=ADMIN,    PASS='$23ADM--', PROTECT-PW=(8, MAX) -
             , KSET=UTMADMIN
USER CLERK   , FORMAT=*BOOK,    PASS='¤$*45jkl', PROTECT-PW=(8, MAX) -
             , KSET=BOOKKEEP
USER PRSNLMNG, FORMAT=*PERSNNL, PASS='78+lsd*/', PROTECT-PW=(8, MAX) -
             , KSET=MPRSNNL
USER MILLER  , FORMAT=*PERSNNL, PASS='7HGFKK*/', PROTECT-PW=( , MED) -
             , KSET=OPRSNNL
USER COMP    , FORMAT=*PERSNNL, PASS='7sdfKK*/', PROTECT-PW=( , MED) -
             , KSET=CPRSNNL
USER CHARGE  , FORMAT=*TRAVEL,  PASS='%aJ1df-+', PROTECT-PW=( , MED) -
             , KSET=OFFCHRG , LOCALE=(EN)
USER CHIEF   , FORMAT=*TRAVEL,  PASS='%aJs5f-+', PROTECT-PW=( , MED) -
             , KSET=OFFADMIN, LOCALE=(EN)
USER TPARIS  , FORMAT=*TRAVEL,  PASS='kj678+*',  PROTECT-PW=( , MED) -
             , KSET=TRVAGNCY, LOCALE=(FR, EU)
USER TNEWYORK, FORMAT=*TRAVEL,  PASS='56asdf$~', PROTECT-PW=( , MED) -
             , KSET=TRVAGNCY, LOCALE=(EN)
```

```
USER TMUNICH , FORMAT=*TRAVEL,  PASS='%as3f$O',  PROTECT-PW=( , MED) -
             , KSET=TRVAGNCY, LOCALE=(DE, EU)
USER TLONDON , FORMAT=*TRAVEL,  PASS='%4Jsdf-+', PROTECT-PW=( , MED) -
             , KSET=TRVAGNCY, LOCALE=(EN)
USER MANOFF  , FORMAT=*BOOK,    PASS='$23ADM--', PROTECT-PW=(8, MAX) -
             , KSET=OFFADMIN

*******************************************
*** PTERM/LTERM-Statements              ***
*******************************************

PTERM PRB22273, LTERM=PRINTER, PRONAM=PRO, PTYPE=T9021, CONNECT=YES
LTERM PRINTER, USAGE=O

PTERM RSO,     LTERM=RSO, PTYPE=*RSO, PRONAM=*RSO, CONNECT=YES
LTERM RSO,     USAGE=O
```

6.7.2 KDCDEF-Anweisungen für UTM-D-Anwendung RMS

```
************************************************************************
*** K D C D E F - S T A T E M E N T S                                ***
*** FOR UTM-D-PROGRAM "RMS"                                         ***
************************************************************************

ROOT RMSROOT

OPTION GEN=ALL

ACCOUNT ACC = YES

FORMSYS

MESSAGE MODULE = KCSMSGS, LOCALE=(EN)
MESSAGE MODULE = MSGSGER, LOCALE=(DE, EU)
MESSAGE MODULE = MSGSFRA, LOCALE=(FR, EU)

MAX LOCALE = (EN)

MAX KDCFILE = (RMS, DOUBLE)  -
   ,APPLINAME = APRMS        -
   ,APPLIMODE = S            -
   ,TASKS = 10               -
   ,ASYNTASKS = 3            -
   ,GSSBS = 200              -
   ,PGPOOL = 2048            -
   ,CACHESIZE = (512,50,RES) -
   ,CONN-USERS = 50          -
   ,RECBUF = (10,1024)       -
   ,KEYVALUE = 20            -
   ,LSSBS = 9                -
   ,LPUTBUF = 10             -
```

```
   ,LPUTLTH = 1948           -
   ,NRCONV = 1               -
   ,TERMWAIT = 600           -
   ,DPUTLIMIT1 = (363,0,0,0) -
   ,DPUTLIMIT2 = (1,0,0,0)   -
   ,KB = 1024                -
   ,NB = 2048                -
   ,SPAB = 4096              -
   ,CLRCH = X'FF'

*********************************************************
*** RESERVE Statement to allow dynamic administration ***
*********************************************************

RESERVE OBJECT=ALL

*********************************************************
*** DATABASE CONTROL Statements                       ***
*********************************************************

DATABASE TYPE=UDS
DATABASE TYPE=ORACLE

*
*
***********************************************
*** SFUNC CONTROL Statements                ***
***********************************************

SFUNC F1 , TAC = INITRESP
SFUNC F2 , TAC = SHUTRESP
SFUNC F5 , TAC = ENQRRESP
SFUNC F6 , TAC = AUTHRESP
SFUNC F7 , TAC = RESRRESP
SFUNC F8 , TAC = AVALRESP
SFUNC F20, TAC = CNCLRESP

***********************************************
*** KSET-Statements                         ***
***********************************************

KSET MASTER  , KEYS=MASTER        "SUPERUSER"
KSET UTMADMIN, KEYS=1             "Administrator of application"
KSET OPRSNNL , KEYS=2             "office personnel   / Büropersonal"
KSET MPRSNNL , KEYS=3             "personnel manager  / Personalchef"
KSET CPRSNNL , KEYS=4             "computer personnel / DV-Mitarbeiter"
KSET OFFCHRG , KEYS=5             "official in charge / Sachbearbeiter"
KSET OFFADMIN, KEYS=6             "administrator of office data"
KSET BOOKKEEP, KEYS=7             "book keeper"
KSET TRVAGNCY, KEYS=8             "travel agencies"
```

```
*********************************************
*** LOAD-MODULE-Statements                ***
*********************************************

LOAD-MODULE BOOKKEEP, VERSION=@, LIB=DYNPROGLIB, LOAD-MODE=STARTUP
LOAD-MODULE PERSNNL , VERSION=@, LIB=DYNPROGLIB, LOAD-MODE=STARTUP
LOAD-MODULE RMS     , VERSION=@, LIB=DYNPROGLIB, LOAD-MODE=STARTUP
LOAD-MODULE OFFICE  , VERSION=@, LIB=DYNPROGLIB, LOAD-MODE=STARTUP

*********************************************
*** EXIT-Statements                       ***
*********************************************

EXIT PROGRAM=HNDLEXIT, USAGE=START
EXIT PROGRAM=HNDLEXIT, USAGE=SHUT

*************************************************************
*** Read data which could be administrated dynamically ***
*************************************************************

* use create-control-statements if application ran before

* CREATE-CONTROL-STATEMENTS *ALL, TO-FILE   = DYNAMIC.RMS.DATA -
*                               , FROM-FILE = COPIED.RMS.KDCA

OPTION DATA=DYNAMIC.RMS

*
*
*
*
*
*

*********************************************
*** TACCLASS-Statements                   ***
*********************************************

* not used

*********************************************
*** TLS-Statements                        ***
*********************************************

TLS TLSA

*********************************************
*** ULS-Statements                        ***
*********************************************

ULS ULSA
ULS ULSB
```

```
*********************************************
*** TPOOL-Statements                        ***
*********************************************

TPOOL LTERM=TP#,   NUMBER=100, PRONAM=*ANY, PTYPE=*ANY, KSET=MASTER
TPOOL LTERM=UPICR, NUMBER=100, PRONAM=*ANY, PTYPE=UPIC-R, KSET=MASTER

*********************************************
*** UTMD-Statements                         ***
*********************************************

UTMD MAXJR = 200, APT=(1,2,3,10), CONCTIME=25, PTCTIME=0

*********************************************
*** Generation of syntax                    ***
*********************************************

ABSTRACT-SYNTAX EUROSI, OBJECT-IDENTIFIER = (1, 3, 9990, 1, 3, 12) -
                      , TRANSFER-SYNTAX = BER

*********************************************
* Generation of APPLICATION CONTEXTS        ***
*********************************************
*
* Without CCR
*
APPLICATION-CONTEXT EUROSIAC, OBJECT-IDENTIFIER = (1, 3, 9990, 1, 4, 12) -
                            , ABSTRACT-SYNTAX = (EUROSI)
*
* Include CCR
*
APPLICATION-CONTEXT EUOSICCR, OBJECT-IDENTIFIER = (1, 3, 9990, 1, 4, 13) -
                            , ABSTRACT-SYNTAX = (EUROSI, CCR)
*
*
*
*
*
*
*
```

```
*********************************************************************
*** OSI TP generation                                             ***
*********************************************************************

*+-----------------------------------------------------------------+
*|                                                                 |
*|           T   R   A   V   E   L   - Connections                 |
*|                                                                 |
*+-----------------------------------------------------------------+

ACCESS-POINT RMS, T-SEL=C'RMS', S-SEL=('SRMS',ASCII) -
               , P-SEL=('PRMS',ASCII), AEQ=1
*
*
*       travel-agency MUNICH  <========>   RMS
OSI-CON MUNICH, LOCAL-ACCESS-POINT=RMS, OSI-LPAP=MUNICH                 -
             , N-SEL=C'HOST0001', T-SEL=C'TRAV', S-SEL=(C'STRV',ASCII) -
             , P-SEL= (C'PTRV',ASCII)
*
*       travel-agency PARIS   <========>   RMS
OSI-CON PARIS , LOCAL-ACCESS-POINT=RMS, OSI-LPAP=PARIS              -
             , N-SEL=C'ISO09', T-SEL=C'TRAV', S-SEL=(C'STRV',ASCII) -
             , P-SEL=(C'PTRV',ASCII)
*
*       travel-agency NEWYORK <========>   RMS
OSI-CON NEWYORK, LOCAL-ACCESS-POINT=RMS, OSI-LPAP=NEWYORK         -
              , N-SEL=C'ISO10', T-SEL=C'TRAV', S-SEL=('2',ASCII) -
              , P-SEL=('2',ASCII)
*
*       travel-agency LONDON  <========>   RMS
OSI-CON LONDON , LOCAL-ACCESS-POINT=RMS, OSI-LPAP=LONDON          -
              , N-SEL=C'ISO06', T-SEL=C'TRAV', S-SEL=('2',ASCII) -
              , P-SEL=('2',ASCII)
*
OSI-LPAP MUNICH , ASS-NAMES=MUNICH, ASSOCIATIONS=4, CONNECT=0  -
                , CONTWIN=0, APPLICATION-CONTEXT=EUOSICCR      -
                , APT=(1,2,3,21),AEQ=1, KSET=TRVAGNCY
OSI-LPAP PARIS  , ASS-NAMES=PARIS, ASSOCIATIONS=4, CONNECT=0   -
                , CONTWIN=0, APPLICATION-CONTEXT=EUOSICCR      -
                , APT=(1,2,3,22), AEQ=1, KSET=TRVAGNCY
OSI-LPAP NEWYORK, ASS-NAMES=NEWYORK, ASSOCIATIONS=1, CONNECT=0 -
                , CONTWIN=0, APPLICATION-CONTEXT=EUOSICCR      -
                , APT=(1,2,3,23), AEQ=1, KSET=TRVAGNCY
OSI-LPAP LONDON , ASS-NAMES=LONDON, ASSOCIATIONS=1, CONNECT=0  -
                , CONTWIN=0, APPLICATION-CONTEXT=EUOSICCR      -
                , APT=(1,2,3,24), AEQ=1, KSET=TRVAGNCY

*
*
```

```
*
*
*
*
*
*
*
*
*+---------------------------------------------------------------------+
*|            From RMS to all servers                                  |
*+---------------------------------------------------------------------+
*
*
*       RMS   <========> Server
*
OSI-LPAP BANK   , ASS-NAMES=BANK, ASSOCIATIONS=4, CONNECT=4   -
                , CONTWIN=4, APPLICATION-CONTEXT=EUOSICCR     -
                , APT=(1,2,3,30), AEQ=1
OSI-LPAP GOLF   , ASS-NAMES=GOLF, ASSOCIATIONS=4, CONNECT=4   -
                , CONTWIN=4, APPLICATION-CONTEXT=EUOSICCR     -
                , APT=(1,2,3,30), AEQ=2
OSI-LPAP HOTEL  , ASS-NAMES=HOTEL, ASSOCIATIONS=4, CONNECT=4  -
                , CONTWIN=4, APPLICATION-CONTEXT=EUOSICCR     -
                , APT=(1,2,3,30), AEQ=3
OSI-LPAP AIRLINE, ASS-NAMES=FLIGHT, ASSOCIATIONS=4, CONNECT=4 -
                , CONTWIN=4,APPLICATION-CONTEXT=EUOSICCR      -
                , APT=(1,2,3,30), AEQ=4
*
*
LTAC BANK, LPAP=BANK,    RTAC=BANK, STATUS=ON, TYPE=D
*
*
OSI-CON BANK   , LOCAL-ACCESS-POINT=RMS, OSI-LPAP=BANK                  -
               , N-SEL=C'HOSTOOO1', T-SEL=C'BANK', S-SEL=('SBNK',ASCII) -
               , P-SEL=(C'PBNK',ASCII)
OSI-CON GOLF   , LOCAL-ACCESS-POINT=RMS, OSI-LPAP=GOLF                  -
               , N-SEL=C'HOSTOOO1', T-SEL=C'GOLF', S-SEL=('SGLF',ASCII) -
               , P-SEL=('PGLF',ASCII)
OSI-CON HOTEL  , LOCAL-ACCESS-POINT=RMS, OSI-LPAP=HOTEL                 -
               , N-SEL=C'HOSTOOO1', T-SEL = C'HOTL'                     -
               , S-SEL = ('SHTL',ASCII), P-SEL = ('PHTL',ASCII)
OSI-CON AIRLINE, LOCAL-ACCESS-POINT=RMS, OSI-LPAP=AIRLINE               -
               , N-SEL=C'HOSTOOO1', T-SEL = C'FLGH'                     -
               , S-SEL=('SFLG',ASCII), P-SEL=('PFLG',ASCII)

END
```

6.7.3 KDCDEF-Inputdatei DynamicTravel für UTM-Anwendung TRAVEL

```
************************************
*    BANK Program                  *
************************************

PROGRAM BANK,     COMP=C, SHARED-OBJECT=BANK

************************************
*    TRAVEL Programs               *
************************************

PROGRAM SIGN1,    COMP=C
PROGRAM SIGN2,    COMP=C
PROGRAM BDTAC,    COMP=C
PROGRAM TRRECEIV, COMP=C, SHARED-OBJECT=TRAVEL
PROGRAM STRTEX,   COMP=C, SHARED-OBJECT=TRAVEL
PROGRAM MMENUE,   COMP=C, SHARED-OBJECT=TRAVEL
PROGRAM TRINFO1,  COMP=C, SHARED-OBJECT=TRAVEL
PROGRAM TRINFO2,  COMP=C, SHARED-OBJECT=TRAVEL
PROGRAM TRINFO3,  COMP=C, SHARED-OBJECT=TRAVEL
PROGRAM TRINFO4,  COMP=C, SHARED-OBJECT=TRAVEL
PROGRAM TRINFO5,  COMP=C, SHARED-OBJECT=TRAVEL
PROGRAM TRINFO6,  COMP=C, SHARED-OBJECT=TRAVEL
PROGRAM CANCEL,   COMP=C, SHARED-OBJECT=TRAVEL
PROGRAM CANCALL,  COMP=C, SHARED-OBJECT=TRAVEL
PROGRAM TRINQU,   COMP=C, SHARED-OBJECT=TRAVEL

**** Administration

PROGRAM KDCADM,   COMP=C
PROGRAM KDCDADM,  COMP=C
PROGRAM KDCPADM,  COMP=C

************************************
*      TACS BANK                   *
************************************

TAC BANK, PROGRAM=BANK, LOCK=5

************************************
*      TACS TRAVEL AGENCY          *
************************************

TAC KDCBADTC, PROGRAM=BDTAC, TYPE=D
TAC KDCSGNTC, PROGRAM=SIGN1, TYPE=D
TAC SIGNON2 , PROGRAM=SIGN2, TYPE=D
TAC MMENUE  , PROGRAM=MMENUE  , LOCK=5
TAC INFO1   , PROGRAM=TRINFO1 , LOCK=5
TAC INFO2   , PROGRAM=TRINFO2 , LOCK=5
TAC INFO3   , PROGRAM=TRINFO3 , LOCK=5
TAC INFO4   , PROGRAM=TRINFO4 , LOCK=5
```

```
TAC INFO5   , PROGRAM=TRINFO5 , LOCK=5
TAC INFO6   , PROGRAM=TRINFO6 , LOCK=5
TAC TRRECEIV, PROGRAM=TRRECEIV, LOCK=5
TAC CANCEL  , PROGRAM=CANCEL  , LOCK=5, CALL=NEXT,  TYPE=D
TAC CANCALL , PROGRAM=CANCALL , LOCK=5, CALL=FIRST, TYPE=D
TAC INQUIRY , PROGRAM=TRINQU  , LOCK=5
*

**** ADMINISTRATION DIALOG ***

TAC KDCTAC  , LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCLTERM, LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCPTERM, LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCSWTCH, LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCSEND , LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCAPPL , LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCUSER , LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCDIAG , LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCLOG  , LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCINF  , LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCHELP , LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCLPAP , LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCLTAC , LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCSHUT , LOCK=1, ADMIN=Y, PROGRAM=KDCADM
TAC KDCTCL  , LOCK=1, ADMIN=Y, PROGRAM=KDCADM

TAC TACDADM  , PROGRAM=KDCDADM, LOCK=1, ADMIN=Y
TAC TACPADM  , PROGRAM=KDCPADM, LOCK=1, ADMIN=Y

*** ADMINISTRATION ASYNCHRON ***

TAC KDCTACA , LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCLTRMA, LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCPTRMA, LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCSWCHA, LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCUSERA, LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCSENDA, LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCAPPLA, LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCDIAGA, LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCLOGA , LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCINFA , LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCHELPA, LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCLPAPA, LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCLTACA, LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCSHUTA, LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
TAC KDCTCLA , LOCK=1, ADMIN=Y, TYPE=A, PROGRAM=KDCADM
*
```

```
*********************************************
*** USER-Statements                       ***
*********************************************

USER SUPERUSR, PERMIT=ADMIN,    PASS='$23ADM--', PROTECT-PW=(8, MAX)        -
             , KSET=MASTER
USER UTMADMIN, PERMIT=ADMIN,    PASS='$23ADM--', PROTECT-PW=(8, MAX)        -
             , KSET=UTMADMIN
USER CHARGE1 , FORMAT=(TRAVEL, EXTEND),PASS='%aJ1df-+', PROTECT-PW=( ,MED)  -
             , KSET=OFFCHRG
USER CHARGE2 , FORMAT=(TRAVEL, EXTEND),PASS='%aJ1df-+', PROTECT-PW=( , MED) -
             , KSET=OFFCHRG
*            .
*********************************************
*** PTERM/LTERM-Statements                ***
*********************************************

PTERM PRINTX,  LTERM=PRINTER, PTYPE=PRINTER, CONNECT=YES
LTERM PRINTER, USAGE=O
```

6.7.4 KDCDEF-Anweisungen für die UTM-Anwendung TRAVEL

```
************************************************************************
*** K D C D E F - S T A T E M E N T S                                ***
*** FOR UTM-PROGRAM "TRAVEL"                                        ***
************************************************************************

ROOT TRAVROOT

OPTION GEN=ALL

FORMSYS

MESSAGE MODULE = KCSMSGS

MAX KDCFILE = (TRAVFILE, DOUBLE) -
   ,APPLINAME = APTRAVEL         -
   ,APPLIMODE = S                -
   ,TASKS = 7                    -
   ,ASYNTASKS = 3                -
   ,GSSBS = 200                  -
   ,PGPOOL = (2048)              -
   ,CACHESIZE = (512,50)         -
   ,CONN-USERS = 50              -
   ,TRACEREC = 30000             -
   ,RECBUF = (10,1024)           -
   ,KEYVALUE = 20                -
   ,LSSBS = 9                    -
   ,LPUTBUF = 10                 -
```

```
   ,LPUTLTH = 1948               -
   ,NRCONV = 1                   -
   ,TERMWAIT = 600         -
   ,DPUTLIMIT1 = (363,0,0,0)     -
   ,DPUTLIMIT2 = (1,0,0,0)       -
   ,KB = 1024                    -
   ,NB = 2048                    -
   ,SPAB = 4096                  -
   ,CLRCH = X'FF'                -
   ,SEMARRAY    =(00001221,5)    -
   ,IPCSHMKEY   = 00012210       -
   ,KAASHMKEY   = 00012220       -
   ,CACHESHMKEY = 00012230       -
   ,OSISHMKEY   = 00012244       -
   ,XAPTPSHMKEY = 00012254

*********************************************************
*** Read data which can be administrated dynamically ***
*********************************************************

* if application ran before use create-control-statements

* CREATE-CONTROL-STATEMENTS *ALL, TO-FILE   = dynamicTravel      -
*                              , FROM-FILE = TRAVFILE/copied.KDCA

OPTION DATA=dynamicTravel

*
*

*********************************************************
*** RESERVE Statement to allow dynamic administration ***
*********************************************************

RESERVE OBJECT=ALL

*********************************************************
***    RMXA                                           ***
*********************************************************

RMXA XASWITCH=xaswitch

***********************************************
*** SHARED-OBJECT Statements               ***
***********************************************

SHARED-OBJECT TRAVEL, LIB=DYNPROGLIB, LOAD-MODE=STARTUP
SHARED-OBJECT BANK,   LIB=DYNPROGLIB, LOAD-MODE=STARTUP
```

```
*********************************************
*** KSET-Statements                         ***
*********************************************

KSET MASTER  , KEYS=MASTER        "SUPERUSER"
KSET UTMADMIN, KEYS=1             "Administrator of application"
KSET OFFCHRG , KEYS=5             "official in charge / Sachbearbeiter"

*********************************************
*** TPOOL-Statements                        ***
*********************************************

TPOOL LTERM=TP#,   NUMBER=100, PTYPE=TTY, KSET=MASTER
TPOOL LTERM=UPICR, NUMBER=100, PTYPE=UPIC-R, KSET=MASTER

*********************************************
*** Generation of syntax                    ***
*********************************************

ABSTRACT-SYNTAX EUROSI, OBJECT-IDENTIFIER = (1, 3, 9990, 1, 3, 12) -
, TRANSFER-SYNTAX = BER

*********************************************
* Generation of APPLICATION CONTEXTS        ***
*********************************************
*
* Without CCR
*
APPLICATION-CONTEXT EUROSIAC, OBJECT-IDENTIFIER = (1, 3, 9990, 1, 4, 12) -
                            , ABSTRACT-SYNTAX = (EUROSI)

*
* Include CCR
*
APPLICATION-CONTEXT EUOSICCR, OBJECT-IDENTIFIER = (1, 3, 9990, 1, 4, 13) -
                            , ABSTRACT-SYNTAX = (EUROSI, CCR)
*
*
*
*

*********************************************************************
*** OSI TP generation                                             ***
*********************************************************************

*********************************************
*** UTMD-Statements                         ***
*********************************************

UTMD MAXJR = 200, APT=(1,2,3,21), CONCTIME=25, PTCTIME=0
```

```
*+-------------------------------------------------------------------+
*|                                                                   |
*|                      R   M   S  -   Connections                   |
*|                                                                   |
*+-------------------------------------------------------------------+

ACCESS-POINT TRAVEL, T-SEL=C'TRAV', S-SEL= (C'STRV',ASCII) -
                   , P-SEL= (C'PTRV',ASCII), AEQ=1 -
                   , LISTENER-PORT=30003, T-PROT=RFC1006 , TSEL-FORMAT=T

*+-------------------------------------------------------------------+
*|            From travel-agency to RMS                             |
*+-------------------------------------------------------------------+
*
*       travel-agency   <========>   RMS
OSI-CON RMS , LOCAL-ACCESS-POINT=TRAVEL, OSI-LPAP=RMS,N-SEL=C'HOST0001'-
            ,T-SEL=C'RMS',S-SEL= (C'SRMS',ASCII), P-SEL= (C'PRMS',ASCII)-
            ,LISTENER-PORT=102, T-PROT=RFC1006, TSEL-FORMAT=T
OSI-LPAP RMS, ASS-NAMES=RMS, ASSOCIATIONS=4, CONNECT=0, CONTWIN=4      -
            , APPLICATION-CONTEXT=EUOSICCR, APT=(1,2,3,10),AEQ=1
*
*
*+-------------------------------------------------------------------+
*|                                                                   |
*|                    B A N K   -   Connections                      |
*|                                                                   |
*+-------------------------------------------------------------------+

SESCHA  PLUC, PLU=Y, PACCNT=0, CONNECT=Y

LPAP LPBANK, SESCHA=PLUC

BCAMAPPL SMP30041                                          -
        ,T-PROT=RFC1006                                    -
        ,LISTENER-PORT=30004,TSEL-FORMAT=T
* Connection 1 for sending ---> BANK---------------*
CON SMP30114,PRONAM=local,BCAMAPPL=SMP30041,LPAP=LPBANK    -
            ,T-PROT=RFC1006                                -
            ,LISTENER-PORT=30001,TSEL-FORMAT=T
LSES SMP30141,RSES=SMP30141,LPAP=LPBANK
* Connection 2 for sending    ---> BANK---------------*
CON SMP30214,PRONAM=local,BCAMAPPL=SMP30041,LPAP=LPBANK    -
            ,T-PROT=RFC1006                                -
            ,LISTENER-PORT=30001,TSEL-FORMAT=T
LSES SMP30241,RSES=SMP30241,LPAP=LPBANK
```

```
* Connection 3 for sending    ---> BANK----------------*
CON SMP30314,PRONAM=local,BCAMAPPL=SMP30041,LPAP=LPBANK    -
            ,T-PROT=RFC1006                                -
            ,LISTENER-PORT=30001,TSEL-FORMAT=T
LSES SMP30341,RSES=SMP30341,LPAP=LPBANK

*-------------------------------------------------------------*
*        LTAC's              -------> BANK
*-------------------------------------------------------------*

LTAC  bank, RTAC=BANK, WAITTIME=(10,30), LPAP=LPBANK
*-------------------------------------------------------------*
*        LTAC's              -------> RMS
*-------------------------------------------------------------*
LTAC AVALRESP, LPAP=RMS
LTAC RESRRESP, LPAP=RMS
LTAC CNCLRESP, LPAP=RMS
LTAC AUTHRESP, LPAP=RMS
LTAC INITRESP, LPAP=RMS
LTAC SHUTRESP, LPAP=RMS
LTAC ENQRRESP, LPAP=RMS

*

END
```

6.8 Meldungen von KDCDEF

Das Generierungstool KDCDEF protokolliert die eingegebenen Parameter auf SYSLST (BS2000/OSD) bzw. *stdout* (Unix-, Windows-Systeme). Darüber hinaus erzeugt KDCDEF Meldungen über den Ablauf des Programms, deren Meldungsnummern zwischen K400 und K549 liegen. Mit Ausnahme der Meldungen K401, K513 und K514 werden alle KDCDEF-Meldungen sowohl auf SYSLST als auch auf SYSOUT bzw. *stderr* und *stdout* ausgegeben. Die Meldungen K401, K513 und K514 werden nur auf SYSOUT bzw. *stderr* ausgegeben.
Unter Unix-Systemen verwendet KDCDEF bei der Ausgabe seiner Meldungen NLS-Meldungskataloge.

Bei Meldungen, die sich auf eine bestimmte fehlerhafte Anweisung beziehen, wird zusätzlich vor der Meldung die fehlerhafte Anweisung ausgegeben. Im openUTM-Handbuch „Meldungen, Test und Diagnose“ sind alle Meldungen aufgelistet. Zu jeder Meldung sind gegebenenfalls zu ergreifende Maßnahmen angegeben.

7 Konfiguration einer Anwendung dynamisch ändern

In diesem Kapitel wird beschrieben, was Sie bei der KDCDEF-Generierung der Anwendung beachten müssen, wenn Sie die Funktionen der dynamischen Konfigurierung in Ihrer Anwendung nutzen wollen. openUTM stellt Ihnen an der Programmschnittstelle KDCADMI sowie über den Administrationsarbeitsplatz WinAdmin Funktionen zur Verfügung, mit denen Sie im Anwendungsbetrieb Objekte in die Konfiguration der Anwendung eintragen bzw. aus der Konfiguration löschen können. Dadurch wird die Verfügbarkeit von UTM-Anwendungen weiter erhöht, da eine Neugenerierung der Anwendung mit KDCDEF, für die der Betrieb unterbrochen werden muss, weitaus seltener erforderlich ist. Voraussetzung für die Nutzung der Funktionen zur dynamischen Konfigurierung ist, dass Sie bei der Generierung mit der KDCDEF-Steueranweisung RESERVE Tabellenplätze in den Objekttabellen von openUTM reservieren.

Dies macht es Ihnen möglich, Services sowie Clients und Drucker mit den zugeordneten LTERM-Partnern dynamisch in die Konfiguration einzutragen und Benutzerkennungen dynamisch zu erzeugen. All diese Objekte können auch dynamisch gelöscht werden.

Im Einzelnen können Sie folgende Objekte dynamisch erzeugen und löschen:

- Transaktionscodes und TAC-Queues
- Teilprogramme und VORGANG-Exits (nur in Anwendungen mit Lademodulen, Shared Objects bzw. DLLs)
- Benutzerkennungen
- LTERM-Partner
- Keysets
- lokale Servicenamen
- Transportverbindungen zu LU6.1-Partner-Anwendungen und LU6.1-Sessionnamen
- Kommunikationspartner des Typs TS-Anwendung, UPIC-Client und Terminal

B/X
- Drucker

Um die Funktionen der dynamischen Konfiguration nutzen zu können, müssen Sie Administrationsprogramme erstellen oder die openUTM-Komponente WinAdmin einsetzen. Mit dem Aufruf KC_CREATE_OBJECT der Programmschnittstelle zur Administration können Sie neue Objekte in die Konfiguration eintragen und mit dem Aufruf KC_DELETE_OBJECT können Sie Objekte aus der Konfiguration löschen. Im openUTM-Handbuch „Anwendungen administrieren“ wird beschrieben, was Sie beim Erstellen von Administrationsprogrammen zum dynamischen Eintragen von Objekten sowie beim Löschen von Objekten aus der Konfiguration der Anwendung beachten müssen.

Die Funktionen der dynamischen Konfigurierung sind auch in der Funktionsvariante UTM-F in vollem Umfang nutzbar. openUTM sichert alle Änderungen der Konfiguration in der KDCFILE. Die geänderten Konfigurationsdaten bleiben dann wie bei UTM-S auch für den nächsten Anwendungslauf verfügbar.

Damit Sie Objekte dynamisch in die Konfiguration Ihrer UTM-Anwendung aufnehmen können, müssen Sie bei der Generierung der Anwendung mit KDCDEF bestimmte Vorbereitungen treffen, siehe Seite 579 und Seite 581.

Für das Löschen von Objekten aus der Konfiguration sind keine Vorbereitungen bei der KDCDEF-Generierung nötig.

7.1 Plätze in den Objekttabellen der KDCFILE reservieren

Die Konfigurationsdaten einer UTM-Anwendung werden in den Objekttabellen der KDCFILE abgelegt, die bei der KDCDEF-Generierung der Anwendung erzeugt wird. Diese Objekttabellen sind nur in dem Maße dynamisch erweiterbar, wie freie Tabellenplätze vorhanden sind. Deshalb müssen Sie für Objekte, die Sie erst im laufenden Betrieb in die Konfiguration der Anwendung aufnehmen wollen, schon bei der Generierung mit der KDCDEF-Anweisung RESERVE freie Tabellenplätze reservieren. Sie können für folgende UTM-Objekte Tabellenplätze reservieren:

UTM-Objekt	**Objekttyp**
Benutzerkennungen	USER
TS-Anwendungen, UPIC-Clients, Terminals und Drucker	PTERM
LTERM-Partner	LTERM
Teilprogramme und VORGANG-Exits	PROGRAM
Transaktionscodes und TAC-Queues	TAC
Transportverbindungen zu LU6.1-Partner-Anwendungen	CON
LU6.1-Sessionnamen	LSES
Keysets	KSET
lokale Servicenamen	LTAC

Mit der RESERVE-Anweisung auf Seite 464 legen Sie fest, wieviele leere Tabellenplätze für einen Objekttyp angelegt werden sollen. Dies entspricht der Anzahl der Einzelobjekte des jeweiligen Objekttyps, die dynamisch konfiguriert werden können.

Die Anzahl der Tabellenplätze, die pro Objekttyp angelegt werden können, ist begrenzt durch die Anzahl der generierbaren Namen. Sehen Sie dazu die Tabelle auf Seite 260.

Die leeren Tabellenplätze in den Objekttabellen werden Objekttyp-spezifisch reserviert, d.h. ein Tabellenplatz, den Sie beispielsweise für einen LTERM-Partner reserviert haben, kann nicht von einem Transaktionscode belegt werden etc.

Während des Anwendungslaufs können Sie genau so viele Objekte eines Typs dynamisch konfigurieren wie leere Tabellenplätze mit KDCDEF reserviert wurden. Durch das Löschen eines anderen Objekts vom gleichen Typ wird nicht unmittelbar ein Tabellenplatz für ein neues Objekt frei.

Eine Ausnahme bilden hier die Benutzerkennungen. Benutzerkennungen können Sie auf zwei Arten löschen, „verzögert“ oder „sofort“. Wird eine Benutzerkennung „verzögert“ gelöscht, dann bleibt der Tabellenplatz belegt (wie bei den anderen Objekttypen). Wird die Benutzerkennung durch „sofortiges“ Löschen aus der Konfiguration herausgenommen, dann wird ihr Tabellenplatz freigegeben. Er kann sofort wieder für eine neue Benutzerkennung verwendet werden.

Bei der Reservierung der Tabellenplätze mit RESERVE ist Folgendes zu beachten: Für jeden UPIC-Client und für jede TS-Anwendung, die Sie dynamisch in die Konfiguration eintragen, erzeugt openUTM intern eine Benutzerkennung. In UTM-Anwendungen, die mit Benutzerkennungen generiert sind (die KDCDEF-Generierung enthält mindestens eine USER-Anweisung), muss deshalb für jeden dynamisch einzutragenden Client vom Typ APPLI, SOCKET, UPIC-R oder UPIC-L zusätzlich ein Tabellenplatz für Benutzerkennungen reserviert werden. Diese Tabellenplätze werden beim Löschen der Clients nicht freigegeben (entspricht dem „verzögerten" Löschen). In Anwendungen ohne Benutzerkennungen werden diese Tabellenplätze von openUTM intern reserviert.

Beispiele

```
RESERVE OBJECT=LTERM, NUMBER=100
```

Das bedeutet, es können bis zu 100 LTERM-Partner dynamisch in die Konfiguration eingetragen werden.

```
RESERVE OBJECT=LTERM, PERCENT=200
```

In diesem Fall wurde die Anzahl der reservierten Tabellenplätze relativ zur Anzahl der statisch generierten LTERM-Partner festgelegt. Es können dynamisch doppelt (200%) so viele LTERM-Partner erzeugt werden, wie bei der KDCDEF-Generierung statisch eingetragen wurden. Wurden bei der KDCDEF-Generierung 50 LTERM-Partner eingetragen, dann können nochmals 100 LTERM-Partner dynamisch eingetragen werden.

```
RESERVE OBJECT=ALL, NUMBER=100
```

Das bedeutet, es können für jeden Objekttyp jeweils 100 Objekte dynamisch eingetragen werden, also 100 Benutzerkennungen, 100 LTERM-Partner usw.

```
RESERVE OBJECT=USER, NUMBER=0
```

Diese Anweisung bewirkt, dass die Anzahl der Objekte des angegebenen Typs (hier USER) dynamisch bis auf den maximal generierbaren Wert erhöht werden kann.

Wegen des hohen Platzbedarfs der Tabellen ist es sinnvoll, einen Wert ≠ 0 für NUMBER anzugeben, um den Speicherverbrauch der Anwendung zu reduzieren.

7.2 Voraussetzungen für das dynamische Eintragen von Objekten

Dieser Abschnitt beschreibt, welche Objekte Sie statisch vorgenerieren müssen und welche Voraussetzungen erfüllt sein müssen, damit Sie Teilprogramme und VORGANG-Exits, Transaktionscodes, Benutzerkennungen sowie LU6.1-Verbindungen dynamisch eintragen können.

Beachten Sie bitte, dass die statisch generierten Grenzwerte auch für dynamisch erzeugte Objekte gelten, z.B. gilt für dynamisch erzeugte Keysets der in MAX ...,KEYVALUE= definierte Wert.

Lockcodes, BCAMAPPL-Namen, Formatierungssystem und LPAP-Partner generieren

In KDCDEF müssen folgende Objekte statisch generiert werden:

- Lockcodes, die Sie den Transaktionscodes und LTERM-Partnern als Zugriffsschutz zuordnen wollen, müssen in dem Bereich zwischen 1 und dem mit MAX ...,KEYVALUE=*number* festgelegten Maximalwert liegen.

  Das Lock-/Keycode-Konzept wird ausführlich im openUTM-Handbuch „Konzepte und Funktionen“ beschrieben.

- Alle Namen der lokalen Anwendung (BCAMAPPL-Namen), über die die Verbindungen zu Clients oder Druckern aufgebaut werden sollen, müssen mit KDCDEF generiert werden. Denken Sie insbesondere daran, dass für die Anbindung von TS-Anwendungen über die Socket-Schnittstelle (PTYPE=SOCKET) eigene BCAMAPPL-Namen generiert werden müssen.

B
B
B
B

- Sollen Benutzerkennungen und LTERM-Partnern Startformate zugeordnet werden, dann muss bei der KDCDEF-Generierung mit der Anweisung FORMSYS ein Formatierungssystem generiert werden. Werden #Formate als Startformate verwendet, dann muss zusätzlich ein Anmelde-Vorgang generiert werden.

- Falls Sie LU6.1-Verbindungen oder Sessionnamen dynamisch eintragen möchten, dann müssen die LPAP-Partner sowie die Sessioneigenschaften (SESCHA-Anweisung) statisch generiert sein.

Voraussetzungen für Teilprogramme und VORGANG-Exits

Neue Teilprogramme und VORGANG-Exits können Sie nur dann dynamisch in die Konfiguration der Anwendung aufnehmen, wenn die UTM-Anwendung folgende Bedingung erfüllt:

B
- *BS2000/OSD*:

  B Die Anwendung wurde mit Lademodulen generiert (KDCDEF-Generierung mit LOAD-
  B MODULE-Anweisungen) und zum Binden und Laden des Anwendungsprogramms wird
  B die Funktionalität des BLS genutzt.

  B Das neue Teilprogramm muss in ein Lademodul gebunden werden, das bei der
  B KDCDEF-Generierung definiert wurde. Dieses Lademodul darf nicht statisch ins An-
  B wendungsprogramm eingebunden sein (LOAD-MODE=STATIC), weil ein solches La-
  B demodul nicht dynamisch ausgetauscht werden kann.

X/W
- *Unix-, Windows-Systeme*:

  X/W Die Anwendung wurde mit Shared Objects bzw. DLLs generiert (KDCDEF-Generie-
  X/W rung mit SHARED-OBJECTS-Anweisungen).
  X/W Das neue Teilprogramm muss in ein Shared Object bzw. DLL gebunden werden, das
  X/W bei der KDCDEF-Generierung definiert wurde.

Für jede Programmiersprache, in der Sie Teilprogramme Ihrer Anwendung erstellen wollen, muss mindestens ein Teilprogramm mit KDCDEF generiert werden. Nur dann sind die für den Ablauf benötigten Sprachanschlussmodule und Laufzeitsysteme im Anwendungsprogramm enthalten.

B
B
B
B
B

BS2000/OSD:
Für Teilprogramme, die mit ILCS-fähigen Compilern (COMP=ILCS) übersetzt werden, genügt es, wenn Sie bei der KDCDEF-Generierung ein Teilprogramm mit COMP=ILCS generieren. Es müssen keine PROGRAM-Anweisungen für die verschiedenen Programmiersprachen abgesetzt werden.

Voraussetzungen für Transaktionscodes

Für die dynamische Konfigurierung von Transaktionscodes müssen Sie Folgendes beachten:

- Transaktionscodes für Teilprogramme, die eine X/Open-Programmschnittstelle nutzen, können nur dynamisch erzeugt werden, wenn bei der KDCDEF-Generierung mindestens ein Transaktionscode für ein X/Open-Teilprogramm konfiguriert wurde (TAC-Anweisung mit API≠KDCS).
- Wollen Sie die Transaktionscodes in TAC-Klassen einteilen, um die Auftragsbearbeitung steuern zu können, dann müssen Sie bei der KDCDEF-Generierung mindestens eine TAC-Klasse erzeugen. Dazu gibt es zwei Möglichkeiten:
  - Sie generieren einen Transaktionscode, für den Sie im Operanden TACCLASS (TAC-Anweisung) eine TAC-Klasse angeben, die daraufhin von KDCDEF implizit erzeugt wird.
  - Wenn Sie die Anwendung **ohne** Prioritätensteuerung betreiben (die Anwendung enthält keine TAC-PRIORITIES-Anweisung), dann können Sie TAC-Klassen erzeugen, indem Sie eine TACCLASS-Anweisung schreiben.

  Haben Sie bei der KDCDEF-Generierung eine TAC-Klasse erzeugt, dann können Sie die Transaktionscodes, die Sie dynamisch eintragen, jeder beliebigen TAC-Klasse zwischen 1 und 16 zuordnen. Die TAC-Klassen werden dann von openUTM implizit erzeugt. Diese TAC-Klassen sind administrierbar.

  Ist die Anwendung **ohne** TAC-PRIORITIES generiert, dann legt openUTM bei implizit erzeugten TAC-Klassen die Prozessanzahl (TASKS) wie folgt fest:
  1 für Dialog-TAC-Klassen (Klasse 1 bis 8)
  und 0 für Asynchron-TAC-Klassen (Klasse 9 bis 16).

  Asynchron-TAC-Klassen (9 bis 16) werden von openUTM jedoch nur erzeugt, wenn Sie bei der Generierung in der MAX-Anweisung ASYNTASKS > 0 gesetzt haben.
- In Anwendungen mit TAC-Klassen **ohne** Prioritätensteuerung können Sie Transaktionscodes, die Teilprogrammläufe mit blockierenden Aufrufen starten, nur dann dynamisch erzeugen, wenn TAC-Klassen mit PGWT=YES (Dialog- und/oder Asynchron-TAC-Klasse) statisch generiert wurden.
  Dialog- und Asynchron-TAC-Klassen mit PGWT=YES müssen also bei der KDCDEF-Generierung explizit mit TACCLASS-Anweisungen generiert werden. Zusätzlich muss MAX TASKS-IN-PGWT > 0 gesetzt werden.
- In Anwendungen **mit** Prioritätensteuerung (mit TAC-PRIORITIES-Anweisung) können Sie Transaktionscodes, die Teilprogrammläufe mit blockierenden Aufrufen starten (TAC ...,PGWT=YES), nur dann dynamisch erzeugen, wenn bei der KDCDEF-Generierung MAX TASKS-IN-PGWT>0 gesetzt wurde.

Voraussetzungen für Benutzerkennungen

Benutzerkennungen können Sie nur dynamisch eintragen, wenn Ihre Anwendung mit Benutzerkennungen generiert wird. Dazu muss Ihre KDCDEF-Generierung mindestens eine USER-Anweisung enthalten. Mindestens eine Benutzerkennung muss mit Administrationsberechtigung konfiguriert werden, damit die Aufrufe zur dynamischen Administration von dieser Benutzerkennung ausgeführt werden können.

B Sollen im Betrieb auch Benutzerkennungen mit Ausweiskarte konfigurierbar sein, dann müssen Sie beim Reservieren der Tabellenplätze mit der RESERVE-Anweisung explizit angeben, wieviel Prozent der Tabellenplätze für Benutzerkennungen mit Ausweiskarte belegt werden können.

B Für Benutzerkennungen mit Ausweiskarte muss bei der KDCDEF-Generierung mit der MAX-Anweisung die Länge der Ausweisinformation statisch generiert werden:
MAX...,CARDLTH=*length*

8 Das Tool KDCUPD - KDCFILE aktualisieren

Mit dem Tool KDCUPD können Sie nach einer Neugenerierung Ihrer UTM-Anwendung wichtige Benutzerdaten und Verwaltungsinformationen der Produktiv-Anwendung von der alten in die neue KDCFILE übernehmen. Darüber hinaus können Sie mit Hilfe von KDCUPD von einer älteren openUTM-Version zur aktuellen, neuen openUTM-Version wechseln, ohne die Daten in der KDCFILE der bisherigen Produktiv-Anwendung zu verlieren.

Welche Daten KDCUPD von der alten in die neue KDCFILE überträgt, ist abhängig von der Variante der UTM-Anwendung:

- UTM-S-Anwendungen

  KDCUPD überträgt Verwaltungs- und Benutzerdaten wie GSSBs, asynchrone Nachrichten, TLS- oder ULS-Bereiche und Vorgangs-spezifische Informationen etc. aus der bisherigen KDCFILE in eine neu generierte KDCFILE. Das Tool KDCUPD prüft bei der Datenübertragung, ob der Eigentümer, das Ziel oder der Erzeuger der Daten in der neuen KDCFILE fehlt oder ob er im bisherigen Anwendungslauf per Administration gelöscht wurde. In diesen Fällen überträgt KDCUPD die Daten nicht und protokolliert dies.

  Bei einem KDCUPD-Lauf werden auch alle verfügbaren RSA-Schlüssel übertragen. Dies gilt sowohl für die aktiven Schlüssel als auch die Schlüssel, die per Administration erzeugt wurden, aber noch nicht aktiviert sind. Dabei gilt:

  – Wurde die alte KDCFILE mit einer openUTM-Version ab V5.2 generiert, dann werden alle RSA-Schlüssel der Verschlüsselungsebenen 1 bis 4 übertragen.
    Es werden nur die verfügbaren Schlüssel übertragen. Existieren im alten KDCFILE in einer Verschlüsselungsebene keine RSA-Schlüssel, wird für diese Ebene nichts übertragen, so dass gegebenenfalls im neuen KDCFILE für diese Verschlüsselungsebene generierte RSA-Schlüssel nicht mit 0 überschrieben werden.
  – Wurde die alte KDCFILE mit einer openUTM-Version < V5.2 generiert, dann werden nur die Schlüssel der Verschlüsselungsebene 1 übertragen.

- UTM-F-Anwendungen

  KDCUPD überträgt nur bestimmte Änderungen in den Verwaltungsdaten:
  – Passwörter und RSA-Schlüssel
  – Versions-Nummern der Lademodule und Locales unter BS2000/OSD
  – Versions-Nummern der Shared Objects unter Unix-Systemen
  – Versions-Nummern der DLLs unter Windows-Systemen.

  B X W

  Benutzerdaten werden **nicht** übertragen.

Mit der KDCUPD-Anweisung TRANSFER können Sie steuern, welche Daten übernommen werden sollen. KDCUPD führt vor der Übertragung automatisch eine Konsistenzprüfung für die KDCFILE-Dateien durch.

Mit der KDCUPD-Anweisung CHECK können Sie die KDCFILE-Datei(en) einer Anwendung auf Vollständigkeit und Konsistenz prüfen, ohne Daten zu übertragen.

Unterstützte Übergänge

Die alte KDCFILE, aus der KDCUPD die Daten in eine neu generierte KDCFILE übernehmen soll, kann von einer openUTM-Version 5.2, 5.3 oder 6.0 stammen. Das Dienstprogramm KDCUPD von UTM 6.0 unterstützt also die folgenden Übergänge:

openUTM V5.2 → openUTM V6.0
openUTM V5.3 → openUTM V6.0
openUTM V6.0 → openUTM V6.0

Bei einem Versionswechsel müssen Sie mit dem KDCDEF der neuen openUTM-Version eine neue KDCFILE erzeugen.

Einen umgekehrten Versionswechsel von einer neueren auf eine ältere openUTM-Version unterstützt KDCUPD nicht.

Voraussetzung für den Ablauf von KDCUPD

Voraussetzung für den Ablauf von KDCUPD ist, dass Sie mit dem Generierungstool KDCDEF eine neue KDCFILE erstellt haben und dass die Anwendung normal beendet wurde (z.B. mit dem Administrationskommando KDCSHUT N, W oder G). Wurde die Anwendung abnormal beendet, dann müssen Sie zunächst einen Warmstart durchführen und anschließend die Anwendung normal beenden.

Änderung von Generierungsparametern

KDCUPD vergleicht die Generierungen der beiden KDCFILEs. Abhängig vom Ergebnis dieser Prüfungen kann KDCUPD eventuell einige Daten nicht übertragen und muss in einigen Fällen die Übertragung ganz ablehnen.

Keine Übertragung

Bei der Generierung der neuen KDCFILE darf der Anwender im Vergleich zur alten KDCFILE grundsätzlich alle Generierungsparameter ändern.

Bei Variante UTM-S wird die ganze Übertragung von KDCUPD abgelehnt, wenn es Unterschiede bei bestimmten Generierungsparametern in der neuen und alten Generierung gibt, weil ein Anwendungsbetrieb mit der KDCFILE zu Fehlern führen würde.

Bei der Variante UTM-S sind dies:

- alte KDCFILE mit Formatierung, neue KDCFILE ohne Formatierung generiert
- unterschiedliche Datenbanksysteme oder verschiedene Anzahl der Datenbanksysteme, soweit sie mit DATABASE-Anweisungen definiert wurden

  In folgenden Fällen wird die Übertragung nicht abgelehnt:

  – In der alten KDCFILE ist kein Datenbanksystem, in der neuen KDCFILE sind ein oder zwei Datenbanksysteme generiert.

  – In der alten KDCFILE ist das Datenbanksystem Oracle mit DATABASE, TYPE=ORACLE, in der neuen KDCFILE mit DATABASE, TYPE=XA generiert.
- in den Generierungen sind die gleichen Datenbanksysteme eingetragen, aber die DATABASE-Anweisungen stehen nicht in der gleichen Reihenfolge.

Bei Anwendungen der Variante UTM-F sind solche Unterschiede für KDCUPD kein Hindernis für die Übertragung.

Eingeschränkte Übertragung

Es gibt Generierungsunterschiede zwischen alter und neuer KDCFILE, die zwar eine Übertragung generell ermöglichen, bei denen aber einzelne Nachrichten oder Datenbereiche nicht übertragen werden können. KDCUPD protokolliert solche Ereignisse nach SYSOUT oder SYSLST (bei BS2000/OSD) bzw. nach *stdout* oder *stderr* (bei Unix- und Windows-Systemen) und setzt die Übertragung in die neue KDCFILE fort.

Beispiele

Ist ein LTERM-Partner nicht mehr in der neuen KDCFILE definiert, kann KDCUPD eventuell vorhandene FPUT-Nachrichten und TLS-Bereiche für diesen LTERM-Partner nicht übernehmen.

Ist der Kommunikationsbereich eines Dialog-Vorgangs größer als die maximale KB-Länge (Operand MAX KB=...) in der neuen KDCFILE, dann lehnt KDCUPD die Übertragung dieses Vorgangs ab.

Allgemein gilt, dass bei der Übertragung mit KDCUPD die Generierungswerte in der neuen KDCFILE maßgebend sind.

VORSICHT!
Sätze in Benutzer-Protokolldateien (USLOG-Dateien) müssen Sie vor dem Neustart der Anwendung sichern, da openUTM nach einem KDCUPD-Lauf die aktuelle USLOG-Dateigeneration wieder vom Anfang an beschreibt.

Benutzerdaten, die KDCUPD immer überträgt

KDCUPD überträgt die folgenden Daten einer KDCFILE immer, d.h. unabhängig von den Angaben in der Steueranweisung TRANSFER:

- Asynchron-Nachrichten in USER-Queues, wenn Benutzer, Erzeuger-LTERM und Erzeuger-USER auch in der neuen KDCFILE existieren.

Benutzerdaten, die KDCUPD optional überträgt

Mit der Steueranweisung TRANSFER können Sie steuern, welche der nachfolgenden Daten KDCUPD in die neue KDCFILE übertragen soll:

- Dialog-Vorgänge, die von einem Terminal oder einer TS-Anwendung vom Typ APPLI gestartet wurden.
- Dialog-Vorgänge, die von einem UPIC-Client gestartet wurden.
- Dialog-Vorgänge, die von einer TS-Anwendung vom Typ SOCKET gestartet wurden.
- Passwörter der Benutzerkennungen sowie - falls generiert - Rest-Gültigkeitsdauer, Mindestwartezeit bis zur nächsten Passwortänderung und Passwort-History
- Sekundärspeicherbereiche GSSB, TLS und ULS
- Ausgabe-Aufträge
- Asynchron-Nachrichten an lokale Asynchron-Vorgänge und TAC-Queues sowie offene Asynchron-Vorgänge
- Asynchron-Nachrichten an lokale Partner

- Asynchron-Nachrichten an entfernte Partner
- Asynchron-Nachrichten an temporäre Queues
- aktuelle Versionsnummern der nachladbaren Objekte (Lademodule unter BS2000/OSD, Shared Objects auf Unix-Systemen, DLLs auf Windows-Systemen)

B
- die Locales von Benutzerkennungen (nur auf BS2000/OSD)

Bei der Übertragung von Vorgängen werden alle Vorgangs-spezifischen Daten übertragen:

- LSSB-Daten
- gesicherte Dialog-Nachrichten
- Kommunikationsbereiche
- Stapel gekellerter Vorgänge

Daten, die KDCUPD nicht überträgt

Von KDCUPD werden nicht übertragen:

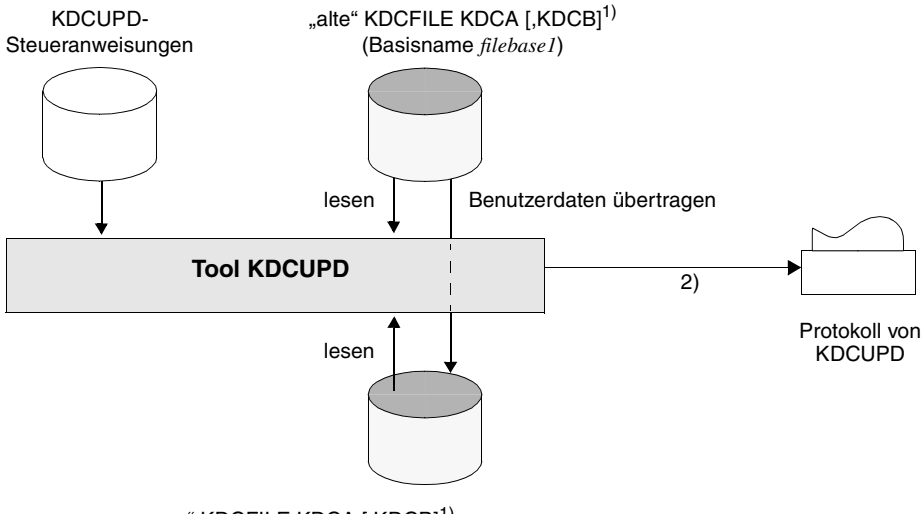
- Objekte, die per Administration neu eingetragen wurden, wie z.B. neue USER.
- Daten, die zu offenen verteilten Vorgängen gehören.
  KDCUPD gibt keine Meldung aus, dass die Daten nicht übertragen wurden!
- Offene Dialog-Vorgänge von Benutzern, wenn der Benutzer in der neuen KDCFILE nicht vorhanden ist.
- Offene Asynchron-Vorgänge, wenn der Benutzer, der den Vorgang gestartet hat, oder der LTERM- oder (OSI-)LPAP-Partner, von dem aus der Asynchron-Vorgang gestartet wurde, in der neuen KDCFILE nicht vorhanden sind.
- Asynchron-Nachrichten, wenn das Ziel der Nachricht oder der Benutzer, der die Nachricht erzeugt hat, oder der LTERM- oder (OSI-)LPAP-Partner, von dem aus die Nachricht erzeugt wurde, in der neuen KDCFILE nicht vorhanden sind.
- Die Speicherbereiche TLS oder ULS, wenn das zugehörige LTERM oder (OSI-)LPAP oder der zugehörige USER oder die zugehörige Session oder Association in der neuen KDCFILE nicht existiert.

8.1 Arbeitsschritte zum Aktualisieren der KDCFILE für standalone UTM-Anwendungen

Anhand der Arbeitsschritte 1 bis 5 können Sie mit dem Tool KDCUPD Ihre KDCFILE aktualisieren:

1. Erzeugen der neuen KDCFILE mit KDCDEF
2. Anwendung normal beenden
3. Alte KDCFILE umbenennen/umkopieren und Benutzerprotokolldatei sichern
4. KDCUPD aufrufen
5. Anwendung starten

Die Arbeitsschritte werden auf den folgenden Seiten ausführlich beschrieben. Die hier geschilderte Methode stellt nur eine von mehreren Möglichkeiten dar. Bei allen Möglichkeiten müssen Sie darauf achten, dass die KDCFILE nicht überschrieben wird, siehe Warnhinweis auf Seite 591.



1)KDCB bei doppelter KDCFILE-Führung

2)KDCUPD-Protokoll;
unter BS2000/OSD nach SYSOUT und SYSLST, unter Unix- und Windows-Systemen nach stdout und stderr

Bild 20: KDCFILE aktualisieren

1. Erzeugen der neuen KDCFILE mit KDCDEF

Sie können mit dem KDCDEF-Lauf die Konfiguration Ihrer Anwendung ändern, d.h. Sie definieren beispielsweise neue Partner-Anwendungen und Verbindungen oder löschen vorhandene oder ändern Anwendungseigenschaften etc.

Der Übergang von einfacher auf doppelte Dateiführung und umgekehrt ist erlaubt. Ebenso kann die „neue“ KDCFILE auf mehrere Dateien aufgeteilt oder die Anzahl dieser Dateien verändert werden. Bei doppelter Dateiführung und/oder Aufteilung der neuen KDCFILE auf mehrere Dateien müssen alle diese Dateien vorhanden und für KDCUPD zugreifbar sein.

Entfallen durch den KDCDEF-Lauf Generierungsbestandteile, wie z.B. Benutzer (USER), TLS-Blöcke, ULS-Blöcke, LTERM-Partner als logische Anschlusspunkte für Clients und Drucker (LTERM), dann kann KDCUPD die dazu gehörenden Daten nicht in die neue KDCFILE übernehmen. Das heißt beispielsweise für entfallene LTERM-Partner, dass die TLS-Bereiche und asynchronen Nachrichten nicht in die neue KDCFILE übernommen werden können. KDCUPD protokolliert die Datenbereiche, die nicht übernommen werden können.

VORSICHT!
Wenn Sie mit KDCDEF die neue KDCFILE erzeugen, müssen Sie darauf achten, dass Sie **nicht versehentlich die alte KDCFILE überschreiben** und wichtige Anwendungsdaten zerstören! Außerdem müssen Sie vor einem Neustart der Anwendung eine eventuell existierende **Benutzerprotokolldatei sichern**, da diese Datei bei einem Neustart nach einem KDCDEF- oder KDCUPD-Lauf überschrieben wird.

Es gibt verschiedene Möglichkeiten, um das Überschreiben der KDCFILE zu vermeiden:

- Sie erzeugen wie hier geschildert die neue KDCFILE schon vor Beendigung des Anwendungslaufs. Dabei verwenden Sie den gleichen Basisnamen, richten aber die KDCFILE in einer anderen Benutzerkennung (BS2000/OSD) bzw. in einem anderen Dateiverzeichnis (Unix-, Windows-Systeme) ein. Nach Beendigung des Anwendungslaufs müssen Sie die Dateien für den nachfolgenden KDCUPD-Lauf ggf. umbenennen oder umkopieren.
- Sie beenden zuerst die Anwendung und benennen dann die alte KDCFILE mit allen zugehörigen Dateien um, indem Sie den Basisnamen ändern. Oder Sie kopieren die alte KDCFILE mit allen Dateien auf eine andere Benutzerkennung (BS2000/OSD) bzw. in ein anderes Dateiverzeichnis (Unix-, Windows-Systeme). Danach starten Sie den KDCDEF-Lauf zum Erzeugen der neuen KDCFILE mit dem bisherigen Basisnamen.

  Beim nachfolgenden KDCUPD-Lauf geben Sie an:

  – KDCFILE OLD= *basisname-umbenannte/umkopierte-KDCFILE*
  – KDCFILE NEW= *basisname-neue-KDCFILE*

- Sie verwenden für die neue KDCFILE einen neuen Basisnamen und arbeiten auch beim KDCUPD-Lauf mit diesem Namen. Beim folgenden Start der Anwendung können Sie entweder den neuen Basisnamen verwenden oder Sie verwenden nach Kopieren und Umbenennen der Dateien wieder den bisherigen Basisnamen.

2. Anwendung normal beenden

Damit die alte KDCFILE in konsistentem Zustand ist, müssen Sie die Anwendung vor dem Aufruf von KDCUPD zunächst normal beenden. Normal beenden können Sie die Anwendung z.B. mit dem Administrationskommando KDCSHUT NORMAL bzw. KDCSHUT WARN oder GRACE (siehe openUTM-Handbuch „Einsatz von openUTM-Anwendungen“).

Beachten Sie, dass beim Beenden der Anwendung die Benutzer-Protokolldatei vorhanden und nicht gesperrt ist. Im Betrieb speichert openUTM die Benutzer-Protokoll-Sätze (LPUT-Aufrufe) zwischen und schreibt sie nicht sofort in die Datei. Beim normalen Beenden der Anwendung wird versucht, diese in die aktuelle Benutzer-Protokolldatei zu schreiben. Gelingt dies nicht, so bleiben die Sätze in der alten KDCFILE.
KDCUPD überträgt diese zwischengespeicherten LPUT-Sätze **nicht** in die neue KDCFILE. KDCUPD zeigt jedoch mit einer Warnung (Meldung K314) an, dass diese Daten verlorengehen.

3. KDCFILE umkopieren und Benutzerprotokolldatei sichern

Nach dem Beenden der Anwendung müssen Sie dann die aktuelle KDCFILE umkopieren bzw. umbenennen, z.B. in OLD.KDCA. Anschließend geben Sie der neuen KDCFILE den 'richtigen' Namen.

Bei doppelter Dateiführung und/oder Aufteilung der aktuellen KDCFILE auf mehrere Dateien müssen alle diese Dateien umkopiert bzw. unter Verwendung desselben Filebase-Namens umbenannt werden und für KDCUPD zugreifbar sein.

Sichern Sie eine evtl. existierende Benutzerprotokolldatei, da die existierende Datei bei einem Neustart nach einem KDCDEF- oder KDCUPD-Lauf überschrieben wird.

4. KDCUPD aufrufen

Beim nachfolgenden KDCUPD-Lauf geben Sie bei der Steueranweisung KDCFILE OLD= den Basisnamen der umkopierten KDCFILE (im BS2000 mit Angabe der UserId) und bei KDCFILE NEW= den Basisnamen der neu erzeugten KDCFILE an.

B *BS2000/OSD:*

B Vor dem Aufruf des Tools KDCUPD müssen Sie Folgendes tun:

B
- Die Bibliothek SYSLNK.UTM.060 als Tasklib zuweisen:

B
```
/SET-TASKLIB LIBRARY=$userid.SYSLNK.UTM.060
```

B B B Es muss die Bibliothek der openUTM-Version zugewiesen werden, mit der auch die neue KDCFILE erstellt wurde. Weisen Sie keine oder eine falsche Bibliothek zu, dann gibt der DLL eine entsprechende Meldung aus.

B
- Prozessschalter 3 auf OFF setzen:

B
```
/MODIFY-JOB-SWITCHES OFF=3
```

B B B KDCUPD wird mit /START-EXECUTABLE-PROGRAM aufgerufen. KDCUPD liest Steueranweisungen von SYSDTA. Die KDCUPD-Steueranweisungen sind ab Seite 597 beschrieben.

B KDCUPD wird wie folgt aufgerufen:

B B B B B B B
```
START-EXECUTABLE-PROGRAM FROM-FILE=*MODULE
                (LIBRARY=$userid.SYSLNK.UTM.060.UTIL,ELEMENT-OR-SYMBOL=KDCUPD)
 :
 :     KDCUPD-Steueranweisungen
 :
 *END
```

B B KDCUPD protokolliert den Ablauf auf SYSLST und/oder SYSOUT (siehe Steueranweisung LIST Seite 600).

X/W *Unix-, Windows-Systeme:*

X/W X/W X/W Vor dem Aufruf von KDCUPD muss sowohl im Basisverzeichnis der alten KDCFILE (*filebase1*) als auch im Basisverzeichnis der neuen KDCFILE (*filebase2*) das Unterverzeichnis DUMP existieren. DUMP wird zu Diagnosezwecken benötigt.

X/W X/W KDCUPD starten Sie unter Unix-Systemen aus der Shell. In Windows-Systemen müssen Sie KDCUPD in einem DOS-Fenster aufrufen. Geben Sie dazu folgendes Kommando ein:

X/W Unix-Systeme: *utmpfad*`/ex/kdcupd 1>upd.out`

W Windows-Systeme: *utmpfad*`\ex\kdcupd 1>upd.out.txt`

X/W Folgendes ist zu beachten:

X/W
X/W
- Das Ausgabeziel *stdout* sollten Sie **immer** auf eine Datei umleiten (hier `upd.out` bzw. `upd.out.txt`).

X/W
X/W
X/W
- *stderr* dürfen Sie **nicht** umlenken (Kommandomodus), da KDCUPD Sie über *stderr* auffordert, die Steueranweisungen einzugeben (z.B. der ‘*‘ des KDCUPD-Eingabemodus wird auf *stderr* ausgegeben).

X/W
X/W
KDCUPD liest die Steueranweisungen von *stdin*. Die Steueranweisungen von KDCUPD sind in Abschnitt „Steueranweisungen für KDCUPD“ auf Seite 597 beschrieben.

X/W
X/W
KDCUPD protokolliert den Ablauf auf *stdout* und *stderr* (siehe Steueranweisung LIST auf Seite 600).

5. Anwendung starten

Im Startparameter FILEBASE= ist der Basisname (*filebase2*) der neuen KDCFILE anzugeben. Ändern Sie Ihre Startprozedur, wenn Sie der neuen KDCFILE beim Erzeugen einen neuen Basisnamen zugeordnet haben.

Wenn mit KDCUPD die Daten in die neue KDCFILE übertragen worden sind und Sie die Anwendung neu starten, kann jeder Benutzer so weiterarbeiten, als wenn die Anwendung normal beendet und anschließend neu gestartet worden wäre.

8.2 Arbeitsschritte zum Aktualisieren der KDCFILE für UTM-Cluster-Anwendungen

Übergang von einer stand-alone UTM-Anwendung auf eine UTM-Cluster-Anwendung V6.0

Beachten Sie beim Übergang von einer stand-alone UTM-Anwendung auf eine UTM-Cluster-Anwendung V6.0 bezüglich des KDCUPD-Laufs Folgendes:

VORSICHT!

Gefahr von Datenverlust.
Wenn Sie sich beim Einrichten und ersten Start einer UTM-Cluster-Anwendung nicht an folgende Vorgaben halten, können Informationen der alten KDCFILE verlorengehen, insbesondere die Information zu den User-Passworten.

- Für UTM-Cluster-Anwendungen wird bei der Generierung eine initiale KDCFILE erstellt. Mit der initialen KDCFILE dürfen Sie nicht starten und Sie dürfen sie auch nicht beim KDCUPD-Lauf als neue KDCFILE angeben.
- Die initiale KDCFILE wird einmal für jede Knoten-Anwendung kopiert.
- Für die KDCFILE der Knoten-Anwendung, die als erstes gestartet werden soll, wird (bei Bedarf) ein KDCUPD-Lauf durchgeführt, d.h. diese KDCFILE wird bei einem KDCUPD-Lauf als neue KDCFILE angegeben. Wenn die User-Passworte aus der alten KDCFILE erhalten bleiben sollen, müssen diese mit dem KDCUPD-Lauf übertragen werden.
- Für die KDCFILEs der Knoten-Anwendungen, die nicht als erste Knoten-Anwendung gestartet werden sollen, darf kein KDCUPD-Lauf durchgeführt werden.
- Die Knoten-Anwendung, für die der KDCUPD-Lauf durchgeführt wurde, muss als erste Knoten-Anwendung gestartet werden.

Neue KDCFILE für bestehende UTM-Cluster-Anwendung

Wenn für eine bestehende UTM-Cluster-Anwendung eine neue KDCFILE erzeugt wird, dann gilt bezüglich des KDCUPD-Laufs Folgendes:

- Die initiale KDCFILE wird einmal für jede Knoten-Anwendung kopiert.
- Vor dem Start jeder Knoten-Anwendung kann für jede Knoten-Anwendung ein KDCUPD-Lauf mit der KDCFILE dieses Knotens als neue KDCFILE durchgeführt werden. Auf diese Weise können z.B. Asynchron-Nachrichten dieser Knoten-Anwendung von der alten KDCFILE in die neue KDCFILE übernommen werden.

Neue Cluster-User-Datei für bestehende UTM-Cluster-Anwendung

Wenn für eine bestehende UTM-Cluster-Anwendung eine neue Cluster-User-Datei erzeugt werden soll, gehen Sie wie folgt vor:

VORSICHT!

Gefahr von Datenverlust.
Wenn Sie sich bei der Neu-Generierung einer Cluster-User-Datei nicht an das nachfolgend beschriebene Vorgehen halten, kann die Information zu allen oder einem Teil der Benutzer-Passworte verlorengehen.

- Um eine neue Cluster-User-Datei einsetzen zu können, müssen Sie, mit Ausnahme einer Knoten-Anwendung, alle Knoten-Anwendungen beenden.
- Vor dem Beenden der letzten Knoten-Anwendung müssen per Administration die aktuellen Werte für alle Benutzer gelesen werden, z.B. über WinAdmin. Dabei werden die User-Passworte aus der Cluster-User-Datei in die KDCFILE dieser Knoten-Anwendung übernommen.
- Mit KDCDEF wird eine neue initiale KDCFILE und eine neue Cluster-User-Datei erstellt.
- Beenden Sie spätestens jetzt auch die letzte Knoten-Anwendung.
- Die initiale KDCFILE wird einmal für jede Knoten-Anwendung kopiert.
- Es wird ein KDCUPD-Lauf durchgeführt um die User-Passworte aus dem alten Anwendungslauf zu übernehmen. Bei diesem KDCUPD-Lauf wird die KDCFILE der als letztes beendeten Knoten-Anwendung als alte KDCFILE angegeben, und die KDCFILE der Knoten-Anwendung, die als erstes gestartet werden soll, wird als neue KDCFILE angegeben.
- Mit der neuen KDCFILE des KDCUPD-Laufs wird die erste Knoten-Anwendung gestartet. Beim Start dieser Knoten-Anwendung werden die User-Passworte aus der KDCFILE in die Cluster-User-Datei übernommen.

8.3 Steueranweisungen für KDCUPD

Für die Datenübertragung kennt KDCUPD folgende Steueranweisungen:

Anweisung	Bedeutung
TRANSFER	Übernahme von Daten der alten KDCFILE in die neue KDCFILE steuern
KDCFILE	Basisname der alten und neuen KDCFILE angeben
B CATID	Unter BS2000/OSD wird die Catid der alten und neuen KDCFILE angegeben
LIST	Ablaufprotokoll steuern
END	Eingabe beenden und Verarbeitung starten

Für die Konsistenzprüfung kennt KDCUPD folgende Steueranweisungen:

Anweisung	Bedeutung
CHECK	KDCFILE einer UTM-Anwendung auf Konsistenz prüfen
LIST	Ablaufprotokoll steuern
END	Eingabe beenden und Verarbeitung starten

Die Steueranweisungen werden unter BS2000/OSD von SYSDTA und unter Unix- und Windows-Systemen von *stdin* (Eingabeaufforderung) gelesen.
Die Steueranweisungen KDCFILE, CATID, LIST und CHECK dürfen Sie pro KDCUPD-Lauf nur einmal eingeben. Die Eingabe muss einzeilig erfolgen. TRANSFER darf mehrmals (einzeilig) angegeben werden.

Beispiel

1. KDCUPD soll die Daten aus der alten KDCFILE (Basisname `BUCH01`) einer UTM-S-Anwendung in die neue KDCFILE (Basisname `BUCH02`) übertragen. Dabei sollen alle Daten übertragen werden mit Ausnahme der an die Kommunikationspartner (LPAP und LTERM) gerichteten Asynchron-Nachrichten, die noch in den Message Queues der alten KDCFILE stehen. KDCUPD soll die positiven Transfer-Meldungen nur nach SYSLST (BS2000/OSD) bzw. *stdout* (Unix-, Windows-Systeme) ausgeben.

Eingaben unter BS2000/OSD	Eingaben auf Unix-/Windows-Systemen

`KDCFILE NEW=BUCH02,OLD=BUCH01`
`CATID OLD=(20SN,20PN),NEW=(20SN,20PN)`
`TRANSFER ASYNLPAP=NO`
`TRANSFER ASYNTERM=NO`
`LIST PROTOCOL=SYSLST`
`END`
	`KDCFILE NEW=BUCH02,OLD=BUCH01`
	
	`TRANSFER ASYNLPAP=NO`
	`TRANSFER ASYNTERM=NO`
	`LIST PROTOCOL=STDOUT`
	`END`

2. Wenn Sie in allen Anweisungen nur die Pflichtparameter angeben, dann versucht KDCUPD, alles zu übertragen und schreibt das Protokoll auf SYSOUT und SYSLST bzw. *stdout* und *stderr*. Um alle Daten zu übertragen, müssen Sie nur folgende Angaben machen:

unter BS2000/OSD	**auf Unix-/Windows-Systemen**

`KDCFILE NEW=BUCH02,OLD=BUCH01`
`CATID OLD=(20SN,20PN),NEW=(20SN,20PN)`
`END`
	`KDCFILE NEW=BUCH02,OLD=BUCH01`
	
	`END`

CATID - Catid für die alte und neue KDCFILE angeben

B Mit der Anweisung CATID wird die Catid der alten und neuen KDCFILE angegeben.

```
B  CATID␣      OLD=(catalog_A,catalog_B)
B              , NEW=(catalog_A,catalog_B)
```

B OLD=(catalog_A,catalog_B)
B Catids für die alte (bisher verwendete) KDCFILE

B NEW=(catalog_A,catalog_B)
B Catids für die neue KDCFILE
B Wenn Sie nur *catalog_A* angeben, wird allen Teilen der KDCFILE diese
B Catid zugeordnet. Wenn Sie mit Catid arbeiten, muss der Basisname
B (*filebase1/2*) in der KDCFILE-Anweisung ohne Catid angegeben werden.

CHECK - KDCFILE auf Konsistenz überprüfen

Mit der Anweisung CHECK prüft KDCUPD die KDCFILE-Datei(en) einer Anwendung auf Konsistenz (Vollständigkeit, gleicher Generierungs- und Verarbeitungsstand). Es werden keine Daten übertragen.

```
CHECK␣      filebase
```

filebase Basisname der KDCFILE
(KDCDEF-Steueranweisung MAX KDCFILE=*filebase*)

B Unter BS2000/OSD muss *filebase* bei doppelter Dateiführung mit der Catid
B der A-Dateien angegeben werden.

END - Eingabe beenden und Verarbeitung starten

Mit der Anweisung END wird die Parametereingabe beendet und die Verarbeitung wird gestartet.
Die Steueranweisungen für KDCUPD müssen mit der Anweisung END abgeschlossen werden.

```
END
```

KDCFILE - Basisname der alten und neuen KDCFILE angeben

Mit der Anweisung KDCFILE wird KDCUPD der Basisname der alten und neuen KDCFILE bekannt gegeben.

```
KDCFILE␣     NEW=filebase2
             , OLD=filebase1
```

NEW=filebase2
: Basisname der neu generierten KDCFILE

B B B B
: *BS2000/OSD:*
Bei doppelter Dateiführung können Sie die Catids für die A- und B-Dateien mit der Anweisung CATID zuweisen. In diesem Fall müssen Sie hier den Basisnamen ohne Catid angeben.

OLD=filebase1
: Basisname der alten KDCFILE

LIST - Ablaufprotokoll steuern

Mit der Anweisung LIST wird die Ausgabe der positiven und negativen Transfermeldungen, sowie der Meldungen K305 und K306 zur Pagepool-Seitenbelegung gesteuert.

```
LIST␣     [ ERRORS={ BOTH | SYSOUT | SYSLST | STDERR | STDOUT} ]
          [ ,INFO={ SHORT | LONG | NO | } ]
          [ ,PROTOCOL={ BOTH | NO | SYSOUT | SYSLST | STDERR | STDOUT } ]
```

B SYSLST und SYSOUT sind nur unter BS2000/OSD erlaubt

X/W STDERR und STDOUT sind nur unter Unix- und Windows-Systemen erlaubt

ERRORS= steuert die Ausgabe der negativen Transfermeldungen

BOTH Die Protokollierung erfolgt unter BS2000/OSD sowohl nach SYSOUT als auch nach SYSLST, unter Unix- und Windows-Systemen sowohl nach *stderr* als auch nach *stdout*.

Standard: BOTH

B SYSOUT Die Protokollierung erfolgt nur nach SYSOUT.

B SYSLST Die Protokollierung erfolgt nur nach SYSLST.

X/W STDOUT Die Protokollierung erfolgt nur nach *stdout*.

X/W STDERR Die Protokollierung erfolgt nur nach *stderr*.

INFO= steuert die Ausgabe der Meldungen K305 und K306 zur Pagepool-Seitenbelegung der neuen KDCFILE

SHORT m Ende des Protokolls der positiven Transfer-Meldungen wird mit K306-Meldungen eine Übersicht der Pagepool-Seitenbelegung ausgegeben. Es wird eine Gesamtübersicht und eine Aufschlüsselung nach den Objekttypen GSSB, QUEUE, LTERM, LPAP, TAC, USER und ASYNVG ausgegeben, sofern sie Pagepool-Seiten belegen.

In die Pagepool-Seitenbelegung geht für die einzelnen Objekttypen ein:

GSSB	Daten des GSSB-Speicherbereichs
QUEUE	Asynchron-Nachrichten der temporären Queue und die zur Administration benötigte Verwaltungsinformation
LTERM	TLS-Blöcke und Asynchron-Nachrichten (inkl. Verwaltungsinformation) des LTERMs
LPAP	TLS-Blöcke und Asynchron-Nachrichten (inkl. Verwaltungsinformation) des LPAPs bzw. OSI-LPAPs
TAC	Asynchron-Nachrichten (inkl. Verwaltungsinformation) des TACs
USER	ULS-Blöcke, Asynchron-Nachrichten (inkl. Verwaltungsinformation) und Dialog-Vorgangsdaten des Benutzers, ULS-Blöcke des LSES bzw. OSI-ASS.
ASYNVG	Vorgangsdaten (inkl. evtl. vorhandener LSSBs) aller offenen Asynchron-Vorgänge des Benutzers

Standard: SHORT

LONG Zusätzlich wird mit einer K305-Meldung für jedes Objekt dieser Typen nach seiner letzten positiven Transfer-Meldung die Pagepool-Seitenbelegung ausgegeben, sofern mindestens eine Pagepool-Seite belegt ist.

NO Die Meldungen K305 und K306 werden abgeschaltet.

PROTOCOL=
steuert die Ausgabe der positiven Transfermeldungen und der Meldungen zur Pagepool-Seitenbelegung (siehe INFO)

BOTH Die Protokollierung erfolgt unter BS2000/OSD sowohl nach SYSOUT als auch nach SYSLST, unter Unix- und Windows-Systemen sowohl nach *stderr* als auch nach *stdout*.

Standard: BOTH

NO Es werden keine positiven Transfermeldungen und keine Meldungen zur Pagepool-Seitenbelegung ausgegeben.

B SYSOUT Die Protokollierung erfolgt nur nach SYSOUT.

B SYSLST Die Protokollierung erfolgt nur nach SYSLST.

X/W STDOUT Die Protokollierung erfolgt nur nach *stdout*.

X/W STDERR Die Protokollierung erfolgt nur nach *stderr*.

TRANSFER - Übertragung der Benutzerdaten steuern

Mit der Anweisung TRANSFER legen Sie fest, welche Benutzerdaten KDCUPD in die neue KDCFILE übertragen soll.

B

```
TRANSFER␣    [ ASYNLPAP=YES | NO ]
             [ ,ASYNTACS=YES | NO ]
             [ ,ASYNTERM=YES | NO ]
             [ ,DIALOGS=YES | NO ]
             [ ,LOCALE=YES | NO ]
             [ ,PASS=YES | NO ]
             [ ,PROG-VER=YES | NO ]
             [ ,QUEUES=YES | NO ]
             [ ,SOCKET-DIALOGS=YES | NO ]
             [ ,STORAGES=YES | NO ]
             [ ,UPIC-DIALOGS=YES | NO ]
```

Wird die TRANSFER-Anweisung weggelassen, führt KDCUPD die Übertragung der Benutzerdaten so durch, als ob bei allen TRANSFER-Operanden der Wert YES angegeben wurde.

ASYNLPAP=

YES Alle noch nicht ausgegebenen asynchronen Nachrichten an Partner-Anwendungen (verteilte Verarbeitung über LU6.1 / OSI TP) werden übertragen.

NO Diese Nachrichten werden nicht übertragen.

ASYNTACS=

YES Alle noch nicht bearbeiteten Hintergrund-Aufträge, auch die zeitgesteuerten, sowie alle offenen Asynchron-Vorgänge mit ihren Daten werden übertragen. Zusätzlich werden sämtliche Nachrichten aller TAC-Queues übernommen.

Nachrichten der Dead Letter Queue werden übernommen, unabhängig davon, ob die ursprünglichen Ziele in der neuen Generierung noch existieren, oder DEAD-LETTER-Q=YES für TACs generiert wurde.

VORSICHT!

Für wrap-around Queues gehen bei Erreichen des Queue Levels die zuerst übertragenen Nachrichten zugunsten der zuletzt übertragenen Nachrichten ohne Meldung verloren.

NO Diese Aufträge und Daten werden nicht übertragen.

Es werden weder Asynchron-Nachrichten noch offene Asynchron-Vorgänge übertragen.

ASYNTERM=

YES Alle noch nicht ausgegebenen Asynchron-Nachrichten an LTERM-Partner werden übertragen, auch die zeitgesteuerten.

NO Diese Nachrichten werden nicht übertragen.

DIALOGS=

YES Die Daten zu Dialog-Vorgängen, die von einem Terminal oder einer TS-Anwendung vom Typ APPLI aus gestartet wurden, werden übertragen. Bei offenem Vorgang sind das LSSBs, KB und die letzte Dialog-Nachricht. Bei beendetem Vorgang ist es die letzte gesicherte Dialog-Nachricht.

NO Diese Daten werden nicht übertragen.

B LOCALE=

B YES KDCUPD überträgt für jeden UTM-Benutzer (USER) die aktuellen Werte seiner Sprachumgebung (Locale) in die neue KDCFILE. Die Werte können von den generierten Werten abweichen, z.B. wenn ein Benutzer seine Sprachumgebung im Anwendungslauf mit dem Aufruf SIGN CL geändert hat.

B NO Die Sprachumgebungen der Benutzer werden nicht übertragen, d.h. es gelten die generierten Werte.

PASS=

YES Es werden Passwörter aus der alten in die neue KDCFILE übernommen. Das gilt für alle USER, für die in der alten und in der neuen KDCFILE ein Passwort generiert wurde. Zusätzlich werden dabei übernommen (falls generiert):
- die verbleibende Gültigkeitsdauer des Passworts
- die zuletzt benutzten Passwörter, also die Passwort-History
- die Mindestzeit bis zur nächsten erlaubten Passwortänderung

Bei Benutzern, für die in der alten KDCFILE (im Gegensatz zur neuen) kein Passwort definiert war, bleibt das neue Passwort erhalten.
Ist für einen Benutzer in der neuen KDCFILE kein Passwort generiert, wird ein in der alten KDCFILE vorhandenes nicht übertragen.

NO Es werden keine Passwörter übernommen.

PROG-VER=

YES Die aktuellen Versionsnummern der Lademodule (BS2000), Shared Objects (Unix-Systeme) bzw. DLLs (Windows-Systeme) werden in die neue KDCFILE übertragen.

NO Die aktuellen Versionsnummern werden nicht übertragen.

QUEUES=

YES Alle Temporären Queues und deren Nachrichten werden von der alten in die neue KDCFILE übertragen.

NO Die Temporären Queues und deren Nachrichten werden nicht übertragen.

SOCKET-DIALOGS=

YES Die Daten zu Dialog-Vorgängen, die von Socket-Partnern gestartet wurden, werden übertragen. Bei einem offenen Vorgang sind das LSSBs, KB und die letzte Dialog-Nachricht. Bei einem beendeten Vorgang ist es die letzte gesicherte Dialog-Nachricht.

NO Die Daten werden nicht übertragen.

STORAGES=

YES Alle UTM-Sekundärspeicherbereiche, also GSSB, TLS und ULS werden übertragen.

NO Die UTM-Sekundärspeicherbereiche werden nicht übertragen.

UPIC-DIALOGS=

YES Die Daten zu Dialog-Vorgängen, die von UPIC-Clients gestartet wurden, werden übertragen. Bei einem offenen Vorgang sind das LSSBs, KB und die letzte Dialog-Nachricht. Bei einem beendeten Vorgang ist es die letzte gesicherte Dialog-Nachricht.

NO Die Daten werden nicht übertragen.

8.4 Ablaufprotokoll und Meldungen von KDCUPD

Das Tool KDCUPD erzeugt ein Ablaufprotokoll, das neben den eingegebenen Parametern noch folgende wichtige Informationen enthält:

- Angaben über die Daten, die übernommen wurden
- Angaben über die Daten, die nicht übernommen werden konnten (diese Meldungen werden mit einem * gekennzeichnet)
- Kurzinformation zur Pagepool-Seitenbelegung

KDCUPD vergleicht die Generierung der neuen und der alten KDCFILE.
Als Ergebnis dieser Prüfungen kann KDCUPD den Transfer einzelner Benutzerdaten ablehnen, weil sie nicht zu den Generierungsoptionen der neuen KDCFILE passen.
Es ist ebenso möglich, dass KDCUPD den Transfer generell ablehnt, weil einzelne Generierungsoptionen der alten und der neuen KDCFILE so verschieden sind, dass ein Anwendungsstart mit der neuen KDCFILE und den übertragenen Daten nicht möglich wäre (siehe auch Seite 587).

Das Ablaufprotokoll wird standardmäßig auf SYSOUT und SYSLST bzw. *stdout* und *stderr* ausgegeben. Die Ausgabe lässt sich mit der LIST-Anweisung steuern.

Im openUTM-Handbuch „Meldungen, Test und Diagnose“ finden Sie die Meldungen von KDCUPD in englischer Sprache und ihre deutsche Übersetzung. Fehlerursachen und Maßnahmen nach Auftreten der Meldung sind ggf. beschrieben.

X/W Unter Unix- und Windows-Systemen verwendet KDCUPD bei der Ausgabe seiner Meldungen NLS-Meldungskataloge.

Verhalten im Fehlerfall

Bei einem internen Fehler erzeugt KDCUPD einen UTM-Dump (unter Unix- und Windows-Systemen finden Sie den Dump im Unterverzeichnis DUMP des Basisverzeichnisses). Dieser Dump kann mit dem Aufbereitungstool KDCDUMP aufbereitet werden (siehe openUTM-Handbuch „Meldungen, Test und Diagnose“).

B Unter BS2000/OSD wird zusätzlich der Prozessschalter 3 gesetzt, wenn sich KDCUPD auf Grund eines Fehlers nicht normal beenden kann. Der Prozessschalter 3 wird ebenfalls gesetzt, wenn durch entfallene Generierungsbestandteile nicht alle Daten in die neue KDCFILE übernommen werden konnten (vgl. Meldung K266), KDCUPD sich aber normal beendet.

B Außerdem ist der Prozessschalter 3 gesetzt, wenn kein KDCUPD-Lauf stattgefunden hat, weil bei der Überprüfung der KDCFILEs ein Fehler festgestellt wurde (vgl. Meldungen K258, K263, K268).

Diagnoseunterlagen

Zu einer Fehlermeldung zum Ablauf von KDCUPD sollten folgende Unterlagen mitgeliefert oder zumindest sichergestellt werden:

- UTM-Dump, falls er erzeugt wurde.
  Bei Speicherengpass kann es beispielsweise vorkommen, dass keine Dump-Datei geschrieben werden kann.
  X
  X
  Für UTM-Anwendungen unter Unix-Systemen muss auch der core-dump sichergestellt werden.
- Protokoll von KDCUPD
- KDCDEF-Steueranweisungen für alte und neue KDCFILE
  (falls dem keine Datenschutzgründe entgegenstehen)
- alte KDCFILE
- neue KDCFILE im Zustand vor KDCUPD-Lauf
  (ersatzweise KDCDEF-Steueranweisungen)

9 Anhang

Dieses Kapitel beschreibt, wie Sie Umsetztabellen zur Code-Konvertierung verwenden und wie Sie diese Tabellen modifizieren können.

Beachten Sie, dass eine Code-Konvertierung nur sinnvoll ist, wenn Sie nur abdruckbare Daten übertragen und keine Binärdaten.

9.1 Code-Konvertierungstabellen in BS2000/OSD

Wird von BS2000 aus mit Partnern über das TCP/IP-Protokoll kommuniziert, dann wird der Partner meist mit dem ASCII- oder dem ISO 8859-1-Code arbeiten, während BS2000 in der Regel mit dem EBCDIC Code arbeitet. Damit eine Kommunikation von BS2000 zu diesen Partnern trotzdem einfach ist, bietet openUTM eine automatische Code-Konvertierung an. Die automatische Code-Konvertierung können Sie bei der KDCDEF-Generierung Partnerspezifisch mit Hilfe des Operanden MAP= in der PTERM- oder TPOOL-Anweisung einschalten. Für die Konvertierung können Sie verschiedene Umsetztabellen verwenden.

Umsetztabellen

Für die Konvertierung können bis zu vier verschiedene Umsetztabellen verwendet werden. Die Tabelle 1 wird bereits vorbelegt ausgeliefert. Sie ist für 7 Bit ASCII vorgesehen (das achte Bit im ASCII-Code wird ignoriert). Die Tabellen 2, 3 und 4 sind noch frei und können vom Anwender belegt werden. Sie werden mit vollen 8 Bit ausgewertet.

Die Tabellen sind im Assembler Modul KDCEA definiert. Die Source von KDCEA befindet sich in der Bibliothek SYSLIB.UTM.060.EXAMPLE. Wenn eine Tabelle geändert werden soll, muss dieses Modul modifiziert werden. Anschließend muss das Modul neu assembliert und per Anweisung INCLUDE-MODULES zum Anwendungs-Programm dazugebunden werden. Diese Anweisung muss vor der Anweisung RESOLVE-BY-AUTO auf die Bibliothek SYSLNK.UTM.060 wirksam werden.

Welche der Tabellen verwendet werden soll, wird beim Generieren des Partners in der PTERM- oder TPOOL-Anweisung beim Operanden MAP= angegeben.

9.2 Code-Konvertierungstabellen in Unix- und Windows-Systemen

X/W Beim Austausch von Nachrichten einer UTM-Anwendung mit einer Partner-Anwendung gibt es die Möglichkeit, dass openUTM eine automatische ASCII-EBCDIC Code-Konvertierung durchführt. Die automatische Code-Konvertierung können Sie bei der KDCDEF-Generierung Partner-spezifisch mit Hilfe des Operanden MAP=SYSTEM in den KDCDEF-Anweisungen PTERM, TPOOL, OSI-CON und SESCHA einschalten.

X/W Bei der Konvertierung verwendet openUTM eine Standard-Tabelle. Diese befindet sich in *utmpfad* unter `src/kcsaeea.c` (Unix-Systeme) bzw. `src\kcsaeea.c` (Windows-Systeme).

9.2.1 Code-Tabelle in Unix-Systemen modifizieren

X Bei UTM-Anwendungen in Unix-Systemen können Sie diese Standard-Tabelle modifizieren. Dazu gehen Sie wie folgt vor:

X 1. Kopieren Sie die Datei `kcsaeea.c` in ein eigenes Dateiverzeichnis.

X 2. Modifizieren Sie die Tabelle wie gewünscht.

X 3. Übersetzen Sie die modifizierte Source-Datei.

X 4. Binden Sie den Workprozess, indem Sie das `.o`-Objekt vor der Bibliothek `libwork.a` angeben. Falls Sie mit der Bibliothek `libwork.so` binden, müssen Sie diese vorher neu erzeugen. Dazu gibt es das Script *utmpfad*`/shsc/stat2dyn`.

9.2.2 Code-Tabelle in Windows-Systemen modifizieren

W W Die Code-Konvertierungstabellen sind in der Bibliothek `utmconvt.dll` enthalten. `utmconvt.dll` befindet sich im selben Verzeichnis wie `libwork.dll`.

W W W Sie können diese Konvertierungstabellen Ihren eigenen Bedürfnissen anpassen, indem Sie die ausgelieferten Source-Dateien ändern und daraus eine modifizierte `utmconvt.dll` erstellen.

W **Source-Dateien zum Erzeugen neuer Code-Tabellen**

W W Das Verzeichnis *utmpfad*`\src` enthält folgende Source-Dateien, die Sie ggf. anpassen müssen:

W
- `kcsaeea.c` und `kcxaent.c`

  W W W W `kcsaeea.c` enthält die Konvertierungstabellen der bisherigen openUTM-Server-Versionen und wird in der mit dem Produkt ausgelieferten `utmconvt.dll` verwendet. `kcxaent.c` enthält die vollständigen Konvertierungstabellen zwischen Windows-Zeichensatz und EBCDIC.

  W W W Diese Dateien sind C-Sourcen und enthalten jeweils zwei Character Arrays der Länge 256. Das eine Array dient zur ASCII-EBCDIC-Konvertierung, das andere zur EBCDIC-ASCII-Konvertierung, siehe Beispiel unten.

W
- utmconvt.def

  W W Definitions-Datei mit EXPORT-Anweisungen. Diese Datei muss nicht modifiziert werden.

W
- `utmconvt.rc` und `resource.h`

  W W W W Ressourcen-Dateien mit Versions- und Copyrightinformationen. Diese Informationen werden angezeigt, wenn man mit der rechten Maustaste die dll-Datei anklickt und *Eigenschaften* auswählt. Diese Dateien müssen nicht unbedingt mit eingebunden werden.

W **Bibliothek utmconvt.dll modifizieren**

W Zum Modifizieren der Bibliothek `utmconvt.dll` sind drei Schritte notwendig:

W 1. Modifizieren Sie die Codetabelle nach Ihren Wünschen (falls erforderlich). Dazu editieren Sie entweder die Datei `kcsaeea.c` oder die Datei `kcxaent.c` mit einem Texteditor. Wenn Sie anstelle der Standard-Konvertierungstabelle `kcsaeea.c` die Windows-Konvertierungstabelle `kcxaent.c` verwenden möchten, können Sie Schritt 1. auslassen.

W *Beispiel*

W Sie möchten die Umlaute in die Konvertierungstabelle `kcsaeea.c` einbauen. Dazu gehen Sie für den Buchstaben 'Ä' wie folgt vor:

W a) Stellen Sie die Codierung von 'Ä' fest. 'Ä' hat in ISO 8859-1 den Code X'C4' (= 196 dezimal) und in EBCDIC.DF.04-1 den Code X'63' (= 99 dezimal).

W b) Ändern Sie den Wert von `kcsaebc[196]` von 0xff nach 0x63 (ASCII-EBCDIC-Konvertierung)

W c) Ändern Sie den Wert von `kcseasc[99]` von 0x1a nach 0xc4 (EBCDIC-ASCII- Konvertierung)

W Für die anderen Umlaute verfahren Sie analog.

W 2. Rufen Sie das Microsoft Visual C++ Developer Studio auf und gehen Sie wie folgt vor:

W - Erstellen Sie ein neues Projekt mit dem Namen `utmconvt` im Verzeichnis `UTMPATH\utmconvt`. Das Projekt muss vom Typ *Dynamic-Link Library* sein.

W - Fügen Sie die folgenden Dateien zum Projekt hinzu:
  - Die eventuell modifizierte Code-Tabellen-Datei (entweder `kcsaeea.c` oder `kcxaent.c`, siehe 1.),
  - `utmconvt.def`
  - und gegebenenfalls `utmconvt.rc`.

W - Erstellen Sie mit diesem Projekt die Bibliothek `utmconvt.dll`.

W - Schließen Sie das Developer Studio.

W 3. Ersetzen Sie die alte Bibliothek `utmconvt.dll` durch die neue Bibliothek:

W - Sichern Sie als Erstes die alte Bibliothek unter einem anderen Namen, damit Sie im Fehlerfall wieder darauf zurückgreifen können.

W - Kopieren Sie die neue `utmconvt.dll` in das Verzeichnis, in dem die UTM-Bibliothek `libwork.dll` steht (in der Regel ist dies *utmpfad*`\ex`). Vergewissern Sie sich, dass die ursprüngliche `utmconvt.dll` dabei auch tatsächlich durch die neue `utmconvt.dll` ersetzt wurde.

W Die neue Konvertierungsbibliothek ist damit einsatzfähig.

Fachwörter

Fachwörter, die an anderer Stelle erklärt werden, sind mit *kursiver* Schrift ausgezeichnet.

Ablaufinvariantes Programm
reentrant program
siehe *reentrant-fähiges Programm*.

Abnormale Beendigung einer UTM-Anwendung
abnormal termination of a UTM application
Beendigung einer *UTM-Anwendung*, bei der die *KDCFILE* nicht mehr aktualisiert wird. Eine abnormale Beendigung wird ausgelöst durch einen schwerwiegenden Fehler, z.B. Rechnerausfall, Fehler in der Systemsoftware. Wird die Anwendung erneut gestartet, führt openUTM einen *Warmstart* durch.

abstrakte Syntax (OSI)
abstract syntax
Eine abstrakte Syntax ist die Menge der formal beschriebenen Datentypen, die zwischen Anwendungen über *OSI TP* ausgetauscht werden sollen. Eine abstrakte Syntax ist unabhängig von der eingesetzten Hardware und der jeweiligen Programmiersprache.

Access-List
access list
Eine Access-List definiert die Berechtigung für den Zugriff auf einen bestimmten *Service*, auf eine bestimmte *TAC-Queue* oder auf eine bestimmte *USER-Queue*. Eine Access-List ist als *Keyset* definiert und enthält einen oder mehrere *Keycodes*, die jeweils eine Rolle in der Anwendung repräsentieren. Benutzer oder LTERMs dürfen nur dann auf den Service oder die *TAC-Queue*/*USER-Queue* zugreifen, wenn ihnen die entsprechenden Rollen zugeteilt wurden, d.h. wenn ihr *Keyset* und die Access-List mindestens einen gemeinsamen *Keycode* enthalten.

Access Point (OSI)
siehe *Dienstzugriffspunkt.*

ACID-Eigenschaften
ACID properties
: Abkürzende Bezeichnung für die grundlegenden Eigenschaften von *Transaktionen*: Atomicity, Consistency, Isolation und Durability.

Administration
administration
: Verwaltung und Steuerung einer *UTM-Anwendung* durch einen *Administrator* oder ein *Administrationsprogramm*.

Administrations-Journal
administration journal
: siehe *Cluster-Administrations-Journal*.

Administrationskommando
administration command
: Kommandos, mit denen der *Administrator* einer *UTM-Anwendung* Administrationsfunktionen für diese Anwendung durchführt. Die Administrationskommandos sind als *Transaktionscodes* realisiert.

Administrationsprogramm
administration program
: *Teilprogramm*, das Aufrufe der *Programmschnittstelle für die Administration* enthält. Dies kann das Standard-Administrationsprogramm *KDCADM* sein, das mit openUTM ausgeliefert wird, oder ein vom Anwender selbst erstelltes Programm.

Administrator
administrator
: Benutzer mit Administrationsberechtigung.

AES
: AES (Advanced Encryption Standard) ist der aktuelle symmetrische Verschlüsselungsstandard, festgelegt vom NIST (National Institute of Standards and Technology), basierend auf dem an der Universität Leuven (B) entwickelten Rijndael-Algorithmus. Wird das AES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen AES-Schlüssel.

Akzeptor (CPI-C)
acceptor
: Die Kommunikationspartner einer *Conversation* werden *Initiator* und Akzeptor genannt. Der Akzeptor nimmt die vom Initiator eingeleitete Conversation mit Accept_Conversation entgegen.

Anmelde-Vorgang (KDCS)
sign-on service
Spezieller *Dialog-Vorgang*, bei dem die Anmeldung eines Clients an eine UTM-Anwendung durch *Teilprogramme* gesteuert wird.

Anschlussprogramm
linkage program
siehe *KDCROOT*.

Anwendungsinformation
application information
Sie stellt die Gesamtmenge der von der *UTM-Anwendung* benutzten Daten dar. Dabei handelt es sich um Speicherbereiche und Nachrichten der UTM-Anwendung, einschließlich der aktuell auf dem Bildschirm angezeigten Daten. Arbeitet die UTM-Anwendung koordiniert mit einem Datenbanksystem, so gehören die in der Datenbank gespeicherten Daten ebenfalls zur Anwendungsinformation.

Anwendungsprogramm
application program
Ein Anwendungsprogramm bildet den Hauptbestandteil einer *UTM-Anwendung*. Es besteht aus der Main Routine *KDCROOT* und den *Teilprogrammen*. Es bearbeitet alle Aufträge, die an eine *UTM-Anwendung* gerichtet werden.

Anwendungs-Wiederanlauf
application restart
Start einer *UTM-S*-Anwendung nach einer vorhergehenden abnormalen Beendigung. Dabei wird die *Anwendungsinformation* auf den zuletzt erreichten konsistenten Zustand gesetzt. Unterbrochene *Dialog-Vorgänge* werden dabei auf den zuletzt erreichten *Sicherungspunkt* zurückgesetzt, so dass die Verarbeitung an dieser Stelle wieder konsistent aufgenommen werden kann (*Vorgangs-Wiederanlauf*). Unterbrochene *Asynchron-Vorgänge* werden zurückgesetzt und neu gestartet oder beim zuletzt erreichten *Sicherungspunkt* fortgesetzt. Bei UTM-F-Anwendungen werden beim Start nach einer vorhergehenden abnormalen Beendigung lediglich die dynamisch geänderten Konfigurationsdaten auf den zuletzt erreichten konsistenten Zustand gesetzt.

Apache Axis
Apache Axis (Apache eXtensible Interaction System) ist eine SOAP-Engine zur Konstruktion von darauf basierenden Web Services und Client-Anwendungen. Es existiert eine Implementierung in C++ und Java.

Apache Tomcat

Apache Tomcat stellt eine Umgebung zur Ausführung von Java-Code auf Web-Servern bereit, die im Rahmen des Jakarta-Projekts der Apache Software Foundation entwickelt wird. Es handelt sich um einen in Java geschriebenen Servlet-Container, der mithilfe des JSP-Compilers Jasper auch JavaServer Pages in Servlets übersetzen und ausführen kann. Dazu kommt ein kompletter HTTP-Server.

Application Context (OSI)

application context

Der Application Context ist die Menge der Regeln, die für die Kommunikation zwischen zwei Anwendungen gelten sollen. Dazu gehören z.B. die abstrakten Syntaxen und die zugeordneten *Transfer-Syntaxen*.

Application Entity (OSI)

application entity

Eine Application Entity (AE) repräsentiert alle für die Kommunikation relevanten Aspekte einer realen Anwendung. Eine Application Entity wird durch einen global (d.h. weltweit) eindeutigen Namen identifiziert, den *Application Entity Title* (AET). Jede Application Entity repräsentiert genau einen *Application Process*. Ein Application Process kann mehrere Application Entities umfassen.

Application Entity Title (OSI)

application entity title

Ein Application Entity Title ist ein global (d.h. weltweit) eindeutiger Name für eine *Application Entity*. Er setzt sich zusammen aus dem *Application Process Title* des jeweiligen *Application Process* und dem *Application Entity Qualifier*.

Application Entity Qualifier (OSI)

application entity qualifier

Bestandteil des *Application Entity Titles*. Der Application Entity Qualifier identifiziert einen *Dienstzugriffspunkt* innerhalb der Anwendung. Ein Application Entity Qualifier kann unterschiedlich aufgebaut sein. openUTM unterstützt den Typ „Zahl“.

Application Process (OSI)

application process

Der Application Process repräsentiert im *OSI-Referenzmodell* eine Anwendung. Er wird durch den *Application Process Title* global (d.h. weltweit) eindeutig identifiziert.

Application Process Title (OSI)
application process title

Gemäß der OSI-Norm dient der Application Process Title (APT) zur global (d.h. weltweit) eindeutigen Identifizierung von Anwendungen. Er kann unterschiedlich aufgebaut sein. openUTM unterstützt den Typ *Object Identifier*.

Application Service Element (OSI)
application service element

Ein Application Service Element (ASE) repräsentiert eine Funktionsgruppe der Anwendungsschicht (Schicht 7) des *OSI-Referenzmodells*.

Association (OSI)
association

Eine Association ist eine Kommunikationsbeziehung zwischen zwei *Application Entities*. Dem Begriff Association entspricht der *LU6.1*-Begriff *Session*.

Asynchron-Auftrag
queued job

Auftrag, der vom Auftraggeber zeitlich entkoppelt durchgeführt wird. Zur Bearbeitung von Asynchron-Aufträgen sind in openUTM *Message Queuing* Funktionen integriert, vgl. *UTM-gesteuerte Queue* und *Service-gesteuerte Queue*. Ein Asynchron-Auftrag wird durch die *Asynchron-Nachricht*, den Empfänger und ggf. den gewünschten Ausführungszeitpunkt beschrieben.
Ist der Empfänger ein Terminal oder Drucker, so ist der Asynchron-Auftrag ein *Ausgabe-Auftrag*; ist der Empfänger ein Asynchron-Vorgang derselben oder einer fernen Anwendung, so handelt es sich um einen *Hintergrund-Auftrag*.
Asynchron-Aufträge können *zeitgesteuerte Aufträge* sein oder auch in einen *Auftrags-Komplex* integriert sein.

Asynchron-Conversation
asynchronous conversation

CPI-C-Conversation, bei der nur der *Initiator* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein asynchroner Transaktionscode generiert sein.

Asynchron-Nachricht
queued message

Asynchron-Nachrichten sind Nachrichten, die an eine *Message Queue* gerichtet sind. Sie werden von der lokalen *UTM-Anwendung* zunächst zwischengespeichert und dann unabhängig vom Auftraggeber weiter verarbeitet. Je nach Empfänger unterscheidet man folgende Typen von Asynchron-Nachrichten:

- Bei Asynchron-Nachrichten an eine *UTM-gesteuerte Queue* wird die Weiterverarbeitung komplett durch openUTM gesteuert. Zu diesem Typ gehören Nachrichten, die einen lokalen oder fernen *Asynchron-Vorgang* starten (vgl. auch *Hintergrund-Auftrag*) und Nachrichten, die zur Ausgabe an ein Terminal oder einen Drucker geschickt wird (vgl. auch *Ausgabe-Auftrag*).
- Bei Asynchron-Nachrichten an eine *Service-gesteuerte Queue* wird die Weiterverarbeitung durch einen *Service* der Anwendung gesteuert. Zu diesem Typ gehören Nachrichten an eine *TAC-Queue*, Nachrichten an eine *USER-Queue* und Nachrichten an eine *Temporäre Queue*. Die User-Queue und die Temporäre Queue müssen dabei zur lokalen Anwendung gehören, die TAC-Queue kann sowohl in der lokalen als auch in einer fernen Anwendung liegen.

Asynchron-Programm
asynchronous program

Teilprogramm, das von einem *Hintergrund-Auftrag* gestartet wird.

Asynchron-Vorgang (KDCS)
asynchronous service

Vorgang, der einen *Hintergrund-Auftrag* bearbeitet. Die Verarbeitung erfolgt entkoppelt vom Auftraggeber. Ein Asynchron-Vorgang kann aus einem oder mehreren Teilprogrammen/Transaktionen bestehen. Er wird über einen asynchronen *Transaktionscode* gestartet.

Auftrag
job

Anforderung eines *Services*, der von einer *UTM-Anwendung* zur Verfügung gestellt wird, durch Angabe eines Transaktionscodes. Siehe auch: *Ausgabe-Auftrag*, *Dialog-Auftrag*, *Hintergrund-Auftrag*, *Auftrags-Komplex*.

Auftraggeber-Vorgang
job-submitting service

Ein Auftraggeber-Vorgang ist ein *Vorgang*, der zur Bearbeitung eines Auftrags einen Service von einer anderen Server-Anwendung (*Auftragnehmer-Vorgang*) anfordert.

Auftragnehmer-Vorgang
job-receiving service

Ein Auftragnehmer-Vorgang ist ein *Vorgang*, der von einem *Auftraggeber-Vorgang* einer anderen Server-Anwendung gestartet wird.

Auftrags-Komplex
job complex

Auftrags-Komplexe dienen dazu, *Asynchron-Aufträgen Quittungsaufträge* zuzuordnen. Ein Asynchron-Auftrag innerhalb eines Auftrags-Komplexes wird *Basis-Auftrag* genannt.

Ausgabe-Auftrag
queued output job

Ausgabeaufträge sind *Asynchron-Aufträge*, die die Aufgabe haben, eine Nachricht, z.B. ein Dokument, an einen Drucker oder an ein Terminal auszugeben. Ausgabeziel kann aber auch eine andere Anwendung sein, die über die Transportsystem-Schnittstelle angeschlossen wurde.
Ausgabeaufträge werden ausschließlich von UTM-Systemfunktionen bearbeitet, d.h. für die Bearbeitung müssen keine Teilprogramme erstellt werden.

Authentisierung
authentication

siehe *Zugangskontrolle*.

Autorisierung
authorization

siehe *Zugriffskontrolle*.

Axis

siehe *Apache Axis*.

Basis-Auftrag
basic job

Asynchron-Auftrag in einem *Auftrags-Komplex*.

Basisformat
basic format

Format, in das der Terminal-Benutzer alle Angaben eintragen kann, die notwendig sind, um einen Vorgang zu starten.

Benutzerausgang
user exit

Begriff ersetzt durch *Event-Exit.*

Benutzerkennung
user ID

Bezeichner für einen Benutzer, der in der *Konfiguration* der *UTM-Anwendung* festgelegt ist (optional mit Passwort zur *Zugangskontrolle*) und dem spezielle Zugriffsrechte (*Zugriffskontrolle*) zugeordnet sind. Ein Terminal-Benutzer oder ein *Client* muss bei der Anmeldung an die UTM-Anwendung diesen Bezeichner (und ggf. das zugeordnete Passwort) angeben. UTM-Anwendungen können auch ohne Benutzerkennungen generiert werden.

Benutzer-Protokolldatei
user log file

Datei oder Dateigeneration, in die der Benutzer mit dem LPUT-Aufruf Sätze variabler Länge schreibt. Jedem Satz werden die Daten aus dem KB-Kopf des *KDCS-Kommunikationsbereichs* vorangestellt. Die Benutzerprotokolldatei unterliegt der Transaktionssicherung von openUTM.

Berechtigungsprüfung
sign-on check

siehe *Zugangskontrolle*.

Beweissicherung (BS2000/OSD)
audit

Im Betrieb einer *UTM-Anwendung* werden zur Beweissicherung sicherheitsrelevante UTM-Ereignisse von *SAT* protokolliert.

Bildschirm-Wiederanlauf
screen restart

Wird ein *Dialog-Vorgang* unterbrochen, gibt openUTM beim *Vorgangswiederanlauf* die *Dialog-Nachricht* der letzten abgeschlossenen *Transaktion* erneut auf dem Bildschirm aus.

Browsen von Asynchron-Nachrichten
browsing asynchronous messages

Ein *Vorgang* liest nacheinander die *Asynchron-Nachrichten*, die sich in einer *Service-gesteuerten Queue* befinden. Die Nachrichten werden während des Lesens nicht gesperrt und verbleiben nach dem Lesen in der Queue. Dadurch ist gleichzeitiges Lesen durch unterschiedliche Vorgänge möglich.

Bypass-Betrieb (BS2000/OSD)
bypass mode

Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Im Bypass-Betrieb wird eine an den Drucker gerichtete *Asynchron-Nachricht* an das Terminal gesendet und von diesem auf den Drucker umgeleitet, ohne auf dem Bildschirm angezeigt zu werden.

Cache-Speicher
cache

Pufferbereich zur Zwischenspeicherung von Anwenderdaten für alle Prozesse einer *UTM-Anwendung*. Der Cache-Speicher dient zur Optimierung der Zugriffe auf den *Pagepool*.

CCS-Name (BS2000/OSD)
CCS name

siehe *Coded-Character-Set-Name*.

Client
client

Clients einer *UTM-Anwendung* können sein:
- Terminals
- UPIC-Client-Programme
- Transportsystem-Anwendungen (z.B. DCAM-, PDN-, CMX-, Socket-Anwendungen oder UTM-Anwendungen, die als *Transportsystem-Anwendung* generiert sind)

Clients werden über LTERM-Partner an die UTM-Anwendung angeschlossen.

openUTM-Clients mit Trägersystem OpenCPIC werden wie *OSI TP-Partner* behandelt.

Client-Seite einer Conversation
client side of a conversation

Begriff ersetzt durch *Initiator*.

Cluster

Eine Anzahl von Rechnern, die über ein schnelles Netzwerk verbunden sind und die von außen in vielen Fällen als ein Rechner gesehen werden können. Das Ziel des „Clustering“ ist meist die Erhöhung der Rechenkapazität oder der Verfügbarkeit gegenüber einem einzelnen Rechner.

Cluster-Administrations-Journal
cluster administration journal

Dateien, in denen administrative Aktionen protokolliert sind, die auf einer *UTM-Cluster-Anwendung* durchgeführt wurden. Die Administrations-Journal-Dateien dienen dazu administrative Aktionen, die in einer UTM-Cluster-Anwendung global auf alle Knoten-Anwendungen wirken sollen, an die anderen Knoten-Anwendungen weiterzugeben.

Cluster-Konfigurationsdatei
cluster configuration file
Datei, die die zentralen Konfigurationsdaten einer *UTM-Cluster-Anwendung* enthält.

Cluster-User-Datei
cluster user file
Datei, die die Verwaltungsdaten der Benutzer einer *UTM-Cluster-Anwendung* enthält.

Coded-Character-Set-Name (BS2000/OSD)
coded character set name
Bei Verwendung des Produkts *XHCS* (e**X**tended **H**ost **C**ode **S**upport) wird jeder verwendete Zeichensatz durch einen Coded-Character-Set-Namen (abgekürzt: „CCS-Name“ oder „CCSN“) eindeutig identifiziert.

Communication Resource Manager
communication resource manager
Communication Resource Manager (CRMs) kontrollieren in verteilten Systemen die Kommunikation zwischen den Anwendungsprogrammen. openUTM stellt CRMs für den internationalen Standard OSI TP, für den Industrie-Standard *LU6.1* und für das openUTM-eigene Protokoll UPIC zur Verfügung.

Contention Loser
contention loser
Jede Verbindung zwischen zwei Partnern wird von einem der Partner verwaltet. Der Partner, der die Verbindung verwaltet, heißt *Contention Winner*. Der andere Partner ist der Contention Loser.

Contention Winner
contention winner
Der Contention Winner einer Verbindung übernimmt die Verwaltung der Verbindung. Aufträge können sowohl vom Contention Winner als auch vom *Contention Loser* gestartet werden. Im Konfliktfall, wenn beide Kommunikationspartner gleichzeitig einen Auftrag starten wollen, wird die Verbindung vom Auftrag des Contention Winner belegt.

Conversation
conversation
Bei CPI-C nennt man die Kommunikation zwischen zwei CPI-C-Anwendungsprogrammen Conversation. Die Kommunikationspartner einer Conversation werden *Initiator* und *Akzeptor* genannt.

Conversation-ID
conversation ID

Jeder *Conversation* wird von CPI-C lokal eine Conversation-ID zugeordnet, d.h. *Initiator* und *Akzeptor* haben jeweils eine eigene Conversation-ID. Mit der Conversation-ID wird jeder CPI-C-Aufruf innerhalb eines Programms eindeutig einer Conversation zugeordnet.

CPI-C

CPI-C (Common Programming Interface for Communication) ist eine von X/Open und dem CIW (**C**PI-C **I**mplementor's **W**orkshop) normierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen. Das in openUTM implementierte CPI-C genügt der CPI-C V2.0 CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung. CPI-C in openUTM kann über die Protokolle OSI TP, LU6.1, UPIC und mit openUTM-LU6.2 kommunizieren.

Cross Coupled System / XCS

Verbund von BS2000-Rechnern mit *Highly Integrated System Complex* Multiple System Control Facility (HIPLEX® MSCF).

Dead Letter Queue
dead letter queue

Die Dead Letter Queue ist eine TAC-Queue mit dem festen Namen KDCDLETQ. Sie steht immer zur Verfügung, um Asynchron-Nachrichten an Transaktionscodes oder TAC-Queues zu sichern, die nicht verarbeitet werden konnten. Die Sicherung von Asynchron-Nachrichten in der Dead Letter Queue kann durch den Parameter DEAD-LETTER-Q der TAC-Anweisung für jedes Nachrichtenziel einzeln ein- und ausgeschaltet werden.

DES

DES (Data Encryption Standard) ist eine internationale Norm zur Verschlüsselung von Daten. Bei diesem Verfahren wird ein Schlüssel zum Ver- und Entschlüsseln verwendet. Wird das DES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen DES-Schlüssel.

Dialog-Auftrag
dialog job, interactive job

Auftrag, der einen *Dialog-Vorgang* startet. Der Auftrag kann von einem *Client* oder - bei *Server-Server-Kommunikation* - von einer anderen Anwendung (Client-Vorgang) erteilt werden.

Dialog-Conversation
dialog conversation

CPI-C-Conversation, bei der sowohl der *Initiator* als auch der *Akzeptor* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein Dialog-Transaktionscode generiert sein.

Dialog-Nachricht
dialog message

Nachricht, die eine Antwort erfordert oder selbst eine Antwort auf eine Anfrage ist. Dabei bilden Anfrage und Antwort einen *Dialog-Schritt*.

Dialog-Programm
dialog program

Teilprogramm, das einen *Dialog-Schritt* teilweise oder vollständig bearbeitet.

Dialog-Schritt
dialog step

Ein Dialog-Schritt beginnt mit dem Empfang einer *Dialog-Nachricht* durch die *UTM-Anwendung*. Er endet mit der Antwort der UTM-Anwendung.

Dialog-Terminalprozess (Unix-/Windows-Systeme)
dialog terminal process

Ein Dialog-Terminalprozess verbindet ein Unix-/Windows-Terminal mit den *Workprozessen* der *UTM-Anwendung*. Dialog-Terminalprozesse werden entweder vom Benutzer durch Eingabe von utmdtp oder über die LOGIN-Shell gestartet. Für jedes Terminal, das an eine UTM-Anwendung angeschlossen werden soll, ist ein eigener Dialog-Terminalprozess erforderlich.

Dialog-Vorgang
dialog service

Vorgang, der einen *Auftrag* im Dialog (zeitlich gekoppelt) mit dem Auftraggeber bearbeitet. Ein Dialog-Vorgang verarbeitet *Dialog-Nachrichten* vom Auftraggeber und erzeugt Dialog-Nachrichten für diesen. Ein Dialog-Vorgang besteht aus mindestens einer *Transaktion.* Ein Dialog-Vorgang umfasst in der Regel mindestens einen *Dialog-Schritt*. Ausnahme: Bei *Vorgangskettung* können auch mehrere Vorgänge einen Dialog-Schritt bilden.

Dienst
service

Programm auf Windows-Systemen, das im Hintergrund unabhängig von angemeldeten Benutzern oder Fenstern abläuft.

Dienstzugriffspunkt
service access point

Im *OSI-Referenzmodell* stehen einer Schicht am Dienstzugriffspunkt die Leistungen der darunterliegenden Schicht zur Verfügung. Der Dienstzugriffspunkt wird im lokalen System durch einen *Selektor* identifiziert. Bei der Kommunikation bindet sich die *UTM-Anwendung* an einen Dienstzugriffspunkt. Eine Verbindung wird zwischen zwei Dienstzugriffspunkten aufgebaut.

Distributed Lock Manager / DLM (BS2000/OSD)

Konkurrierende, Rechner-übergreifende Dateizugriffe können über den Distributed Lock Manager synchronisiert werden.
DLM ist eine Basisfunktion von HIPLEX® MSCF.

Distributed Transaction Processing

X/Open-Architekturmodell für die transaktionsorientierte *verteilte Verarbeitung*.

Druckadministration
print administration

Funktionen zur *Drucksteuerung* und Administration von *Ausgabeaufträgen*, die an einen Drucker gerichtet sind.

Druckerbündel
printer pool

Mehrere Drucker, die demselben *LTERM-Partner* zugeordnet sind.

Druckergruppe (Unix-Systeme)
printer group

Die Unix-Plattform richtet für jeden Drucker standardmäßig eine Druckergruppe ein, die genau diesen Drucker enthält. Darüber hinaus lassen sich mehrere Drucker einer Druckergruppe, aber auch ein Drucker mehreren Druckergruppen zuordnen.

Druckerprozess (Unix-Systeme)
printer process

Prozess, der vom *Mainprozess* zur Ausgabe von *Asynchron-Nachrichten* an eine *Druckergruppe* eingerichtet wird. Er existiert, solange die Druckergruppe an die *UTM-Anwendung* angeschlossen ist. Pro angeschlossener Druckergruppe gibt es einen Druckerprozess.

Druckersteuerstation
printer control terminal

Begriff ersetzt durch *Druckersteuer-LTERM*

Druckersteuer-LTERM
printer control LTERM

Über ein Druckersteuer-LTERM kann sich ein *Client* oder ein Terminal-Benutzer an eine *UTM-Anwendung* anschließen. Von dem Client-Programm oder Terminal aus kann dann die *Administration* der Drucker erfolgen, die dem Druckersteuer-LTERM zugeordnet sind. Hierfür ist keine Administrationsberechtigung notwendig.

Drucksteuerung
print control

openUTM-Funktionen zur Steuerung von Druckausgaben.

Dynamische Konfigurierung
dynamic configuration

Änderung der *Konfiguration* durch die Administration. Im laufenden Betrieb der Anwendung können UTM-Objekte wie z.B. *Teilprogramme*, *Transaktionscodes*, *Clients*, *LU6.1-Verbindungen*, Drucker oder *Benutzerkennungen* in die Konfiguration aufgenommen, modifiziert oder teilweise auch gelöscht werden. Hierzu kann das Administrationsprogramm WinAdmin verwendet werden, oder es müssen eigene *Administrationsprogramme* erstellt werden, die die Funktionen der *Programmschnittstelle der Administration* nutzen.

Einschritt-Transaktion
single-step transaction

Transaktion, die genau einen *Dialog-Schritt* umfasst.

Einschritt-Vorgang
single-step service

Dialog-Vorgang, der genau einen *Dialog-Schritt* umfasst.

Ereignisgesteuerter Vorgang
event-driven service

Begriff ersetzt durch *Event-Service*.

Event-Exit
event exit

Routine des *Anwendungsprogramms*, das bei bestimmten Ereignissen (z.B. Start eines Prozesses, Ende eines Vorgangs) automatisch gestartet wird. Diese darf - im Gegensatz zu den *Event-Services* - keine KDCS-, CPI-C- und XATMI-Aufrufe enthalten.

Event-Funktion
event function

Oberbegriff für *Event-Exits* und *Event-Services*.

Event-Service
event service
Vorgang, der beim Auftreten bestimmter Ereignisse gestartet wird, z.B. bei bestimmten UTM-Meldungen. Die *Teilprogramme* ereignisgesteuerter Vorgänge müssen KDCS-Aufrufe enthalten.

Generierung
generation
Statische Konfigurierung einer *UTM-Anwendung* mit dem UTM-Tool KDCDEF und Erzeugen des Anwendungsprogramms.

Globaler Sekundärer Speicherbereich/GSSB
global secondary storage area
siehe *Sekundärspeicherbereich.*

Globale Transaktion
global transaction
Transaktion, die sich über mehr als eine Anwendung erstreckt.

Hardcopy-Betrieb
hardcopy mode
Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Dabei wird eine Nachricht, die auf dem Bildschirm angezeigt wird, zusätzlich auf dem Drucker abgedruckt.

Heterogene Kopplung
heterogeneous link
Bei *Server-Server-Kommunikation:* Kopplung einer *UTM-Anwendung* mit einer Nicht-UTM-Anwendung, z.B. einer CICS- oder TUXEDO-Anwendung.

Highly Integrated System Complex / HIPLEX®
Produktfamilie zur Realisierung eines Bedien-, Last- und Verfügbarkeitsverbunds mit mehreren BS2000-Servern.

Hintergrund-Auftrag
background job
Hintergrund-Aufträge sind *Asynchron-Aufträge*, die an einen *Asynchron-Vorgang* der eigenen oder einer fernen Anwendung gerichtet sind. Hintergrund-Aufträge eignen sich besonders für zeitintensive oder zeitunkritische Verarbeitungen, deren Ergebnis keinen direkten Einfluss auf den aktuellen Dialog hat.

HIPLEX® MSCF

(MSCF = **M**ultiple **S**ystem **C**ontrol **F**acility)
stellt bei HIPLEX® die Infrastruktur sowie Basisfunktionen für verteilte Anwendungen bereit.

Homogene Kopplung

homogeneous link

Bei *Server-Server-Kommunikation*: Kopplung von *UTM-Anwendungen.* Dabei spielt es keine Rolle, ob die Anwendungen auf der gleichen oder auf unterschiedlichen Betriebssystem-Plattformen ablaufen.

Inbound-Conversation (CPI-C)

inbound conversation

siehe *Incoming-Conversation.*

Incoming-Conversation (CPI-C)

incoming conversation

Eine *Conversation*, bei der das lokale CPI-C-Programm *Akzeptor* ist, heißt Incoming-Conversation. In der X/Open-Specification wird für Incoming-Conversation auch das Synonym Inbound-Conversation verwendet.

Initiale KDCFILE

initial KDCFILE

In einer *UTM-Cluster-Anwendung* die *KDCFILE*, die von *KDCDEF* erzeugt wurde und vor dem Start der Knoten-Anwendungen für jeden Knoten kopiert werden muss.

Initiator (CPI-C)

initiator

Die Kommunikationspartner einer *Conversation* werden Initiator und *Akzeptor* genannt. Der Initiator baut die Conversation mit den CPI-C-Aufrufen Initialize_Conversation und Allocate auf.

Insert

insert

Feld in einem Meldungstext, in das openUTM aktuelle Werte einträgt.

Inverser KDCDEF

inverse KDCDEF

Funktion, die aus den Konfigurationsdaten der *KDCFILE*, die im laufenden Betrieb dynamisch angepasst wurde, Steueranweisungen für einen *KDCDEF*-Lauf erzeugt. Der inverse KDCDEF kann „offline“ unter KDCDEF oder „online“ über die *Programmschnittstelle zur Administration* gestartet werden.

JDK

Java Development Kit
Standard-Entwicklungsumgebung von Sun Microsystems für die Entwicklung von Java-Anwendungen.

Kaltstart
cold start

Starten einer *UTM-Anwendung* nach einer *normalen Beendigung* der Anwendung oder nach einer Neugenerierung (vgl. auch *Warmstart*).

KDCADM

Standard-Administrationsprogramm, das zusammen mit openUTM ausgeliefert wird. KDCADM stellt Administrationsfunktionen zur Verfügung, die über Transaktionscodes (*Administrationskommandos*) aufgerufen werden.

KDCDEF

UTM-Tool für die *Generierung* von *UTM-Anwendungen*. KDCDEF erstellt anhand der Konfigurationsinformationen in den KDCDEF-Steueranweisungen die UTM-Objekte *KDCFILE* und die ROOT-Tabellen-Source für die Main Routine *KDCROOT*.

KDCFILE

Eine oder mehrere Dateien, die für den Ablauf einer *UTM-Anwendung* notwendige Daten enthalten. Die KDCFILE wird mit dem UTM-Generierungstool *KDCDEF* erstellt. Die KDCFILE enthält unter anderem die *Konfiguration* der Anwendung.

KDCROOT

Main Routine eines *Anwendungsprogramms*, die das Bindeglied zwischen *Teilprogrammen* und UTM-Systemcode bildet. KDCROOT wird zusammen mit den *Teilprogrammen* zum *Anwendungsprogramm* gebunden.

KDCS-Parameterbereich
KDCS parameter area

siehe *Parameterbereich*.

KDCS-Programmschnittstelle
KDCS program interface

Universelle UTM-Programmschnittstelle, die den nationalen Standard DIN 66 265 erfüllt und Erweiterungen enthält. Mit KDCS (Kompatible Datenkommunikationsschnittstelle) lassen sich z.B. Dialog-Services erstellen und *Message Queuing* Funktionen nutzen. Außerdem stellt KDCS Aufrufe zur *verteilten Verarbeitung* zur Verfügung.

Kerberos

Kerberos ist ein standardisiertes Netzwerk-Authentisierungsprotokoll (RFC1510), das auf kryptographischen Verschlüsselungsverfahren basiert, wobei keine Kennwörter im Klartext über das Netzwerk gesendet werden.

Kerberos-Principal

Kerberos principal

Eigentümer eines Schlüssels.
Kerberos arbeitet mit symmetrischer Verschlüsselung, d.h. alle Schlüssel liegen an zwei Stellen vor, beim Eigentümer eines Schlüssels (Principal) und beim KDC (Key Distribution Center).

Keycode

key code

Code, der in einer Anwendung eine bestimmte Zugriffsberechtigung oder eine bestimmte Rolle repräsentiert. Mehrere Keycodes werden zu einem *Keyset* zusammengefasst.

Keyset

key set

Zusammenfassung von einem oder mehrerer *Keycodes* unter einem bestimmten Namen. Ein Keyset definiert Berechtigungen im Rahmen des verwendeten Berechtigungskonzepts (Lock-/Keycode-Konzept oder *Access-List*-Konzept). Ein Keyset kann einer *Benutzerkennung*, einem *LTERM-Partner*, einem *Service* oder einer *TAC-Queue* zugeordnet werden.

Knoten

node

Einzelner Rechner eines *Clusters*.

Knoten-Anwendung

node application

UTM-Anwendung, die als Teil einer *UTM-Cluster-Anwendung* auf einem einzelnen *Knoten* zum Ablauf kommt.

Kommunikationsbereich/KB (KDCS)

communication area

Transaktionsgesicherter KDCS-*Primärspeicherbereich*, der Vorgangs-spezifische Daten enthält. Der Kommunikationsbereich besteht aus 3 Teilen:

- dem KB-Kopf mit allgemeinen Vorgangsdaten,
- dem KB-Rückgabebereich für Rückgaben nach KDCS-Aufrufen
- dem KB-Programmbereich zur Datenübergabe zwischen UTM-Teilprogrammen innerhalb eines *Vorgangs*.

Konfiguration
configuration

Summe aller Eigenschaften einer *UTM-Anwendung*. Die Konfiguration beschreibt:

- Anwendungs- und Betriebsparameter
- die Objekte der Anwendung und die Eigenschaften dieser Objekte. Objekte sind z.B. *Teilprogramme* und *Transaktionscodes*, Kommunikationspartner, Drucker, *Benutzerkennungen*
- definierte Zugriffsschutz- und Zugangsschutzmaßnahmen

Die Konfiguration einer UTM-Anwendung wird bei der Generierung festgelegt und kann per *Administration* dynamisch (während des Anwendungslaufs) geändert werden. Die Konfiguration ist in der *KDCFILE* abgelegt.

Konfigurierung
configuration

Festlegen der *Konfiguration* der UTM-Anwendung. Es wird unterschieden zwischen *statischer* und *dynamischer Konfigurierung*.

Logische Verbindung
virtual connection

Zuordnung zweier Kommunikationspartner.

Log4j

Log4j ist ein Teil des Apache Jakarta Projekts. Log4j bietet Schnittstellen zum Protokollieren von Informationen (Ablauf-Informationen, Trace-Records,...) und zum Konfigurieren der Protokoll-Ausgabe. *WS4UTM* verwendet das Softwareprodukt Log4j für die Trace- und Logging-Funktionalität.

Lockcode

Code, um einen LTERM-Partner oder einen Transaktionscode vor unberechtigtem Zugriff zu schützen. Damit ist ein Zugriff nur möglich, wenn das *Keyset* des Zugreifenden den passenden *Keycode* enthält (Lock-/Keycode-Konzept).

Lokaler Sekundärer Speicherbereich/LSSB
local secondary storage area

siehe *Sekundärspeicherbereich*.

LPAP-Bündel
LPAP bundle

LPAP-Bündel ermöglichen die Verteilung von Nachrichten an LPAP-Partner auf mehrere Partner-Anwendungen. Soll eine UTM-Anwendung sehr viele Nachrichten mit einer Partner-Anwendung austauschen, kann es für die Lastverteilung sinnvoll sein, mehrere Instanzen der Partner-Anwendung zu starten und die Nachrichten auf die einzelnen Instanzen zu verteilen. In einem LPAP-Bündel übernimmt *open*UTM die Verteilung der Nachrichten an die Instanzen der Partner-Anwendung. Ein LPAP-Bündel besteht aus einem Master-LPAP und mehreren Slave-LPAPs. Die Slave-LPAPs werden dem Master-LPAP bei der Generierung zugeordnet. LPAP-Bündel gibt es sowohl für das OSI TP-Protokoll als auch für das LU6.1-Protokoll.

LPAP-Partner
LPAP partner

Für die *verteilte Verarbeitung* über das *LU6.1*-Protokoll muss in der lokalen Anwendung für jede Partneranwendung ein LPAP-Partner konfiguriert werden. Der LPAP-Partner spiegelt in der lokalen Anwendung die Partneranwendung wider. Bei der Kommunikation wird die Partneranwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten LPAP-Partners angesprochen.

LTERM-Bündel
LTERM bundle

Ein LTERM-Bündel (Verbindungsbündel) besteht aus einem Master-LTERM und mehreren Slave-LTERMs. Mit einem LTERM-Bündel (Verbindungsbündel) verteilen Sie asynchrone Nachrichten an eine logische Partner-Anwendung gleichmäßig auf mehrere parallele Verbindungen.

LTERM-Gruppe
LTERM group

Eine LTERM-Gruppe besteht aus einem oder mehreren Alias-LTERMs, den Gruppen-LTERMs, und einem Primary-LTERM. In einer LTERM-Gruppe ordnen Sie mehrere LTERMs einer Verbindung zu.

LTERM-Partner
LTERM partner

Um *Clients* oder Drucker an eine *UTM-Anwendung* anschließen zu können, müssen in der Anwendung LTERM-Partner konfiguriert werden. Ein Client oder Drucker kann nur angeschlossen werden, wenn ihm ein LTERM-Partner mit entsprechenden Eigenschaften zugeordnet ist. Diese Zuordnung wird i.A. bei der Konfigurierung festgelegt, sie kann aber auch dynamisch über Terminal-Pools erfolgen.

LTERM-Pool
LTERM pool
: Statt für jeden *Client* eine LTERM- und eine PTERM-Anweisung anzugeben, kann mit der Anweisung TPOOL ein Pool von LTERM-Partnern definiert werden. Schließt sich ein Client über einen LTERM-Pool an, wird ihm dynamisch ein LTERM-Partner aus dem Pool zugeordnet.

LU6.1
: Geräteunabhängiges Datenaustauschprotokoll (Industrie-Standard) für die transaktionsgesicherte *Server-Server-Kommunikation*.

LU6.1-LPAP-Bündel
LU6.1-LPAP bundle
: *LPAP-Bündel* für *LU6.1*-Partner-Anwendungen.

Mainprozess (Unix-/Windows-Systeme)
main process
: Prozess, der die *UTM-Anwendung* startet. Er startet die *Workprozesse*, die *Druckerprozesse, Netzprozesse* und den *Timerprozess* und überwacht die *UTM-Anwendung*.

Main Routine KDCROOT
main routine KDCROOT
: siehe *KDCROOT*.

Mapped Hostname
mapped host name
: Abbildung des UTM-Hostnamen der Partner-Anwendung in einen realen Hostnamen oder umgekehrt.

Meldung / UTM-Meldung
UTM message
: Meldungen werden von dem Transaktionsmonitor openUTM oder von UTM-Tools (wie z.B. *KDCDEF*) an *Meldungsziele* ausgegeben. Eine Meldung besteht aus einer Meldungsnummer und dem Meldungstext, der ggf. *Inserts* mit aktuellen Werten enthält. Je nach Meldungsziel werden entweder die gesamte Meldung oder nur Teile der Meldung (z.B. nur die Inserts) ausgegeben.

Meldungsdefinitionsdatei
message definition file
: Die Meldungsdefinitionsdatei wird mit openUTM ausgeliefert und enthält standardmäßig die UTM-Meldungstexte in deutscher und englischer Sprache und die Definitionen der Meldungseigenschaften. Aufbauend auf diese Datei kann der Anwender auch eigene, individuelle Meldungsmodule erzeugen.

Meldungsziel
message destination

Ausgabemedium für eine *Meldung*. Mögliche Meldungsziele einer Meldung sind z.B. Terminals, *TS-Anwendungen*, der *Event-Service* MSGTAC, die *System-Protokolldatei* SYSLOG oder *TAC-Queues, Asynchron-TACs* und *USER-Queues*.

Mehrschritt-Transaktion
multi-step transaction

Transaktion, die aus mehr als einem *Verarbeitungsschritt* besteht.

Mehrschritt-Vorgang (KDCS)
multi-step service

Vorgang, der in mehreren *Dialog-Schritten* ausgeführt wird.

Message Queuing
message queuing

Message Queuing (MQ) ist eine Form der Kommunikation, bei der die Nachrichten (Messages) nicht unmittelbar, sondern über zwischengeschaltete *Message Queues* ausgetauscht werden. Sender und Empfänger können zeitlich und räumlich entkoppelt ablaufen, die Übermittlung der Nachricht wird garantiert, unabhängig davon, ob gerade eine Netzverbindung besteht oder nicht. Bei openUTM gibt es *UTM-gesteuerte Queues* und *Service-gesteuerte Queues*.

Message Queue
message queue

Warteschlange, in der bestimmte Nachrichten transaktionsgesichert bis zur Weiterverarbeitung eingereiht werden. Je nachdem, wer die Weiterverarbeitung kontrolliert, unterscheidet man *Service-gesteuerte Queues* und *UTM-gesteuerte Queues*.

MSGTAC
MSGTAC

Spezieller Event-Service, der Meldungen mit dem Meldungsziel MSGTAC per Programm verarbeitet. MSGTAC ist ein Asynchron-Vorgang und wird vom Betreiber der Anwendung erstellt.

Multiplexanschluss (BS2000/OSD)
multiplex connection

Spezielle Möglichkeit, Terminals an eine *UTM-Anwendung* anzuschließen. Ein Multiplexanschluss ermöglicht es, dass sich mehrere Terminals eine *Transportverbindung* teilen.

Nachrichten-Bereich/NB (KDCS)
KDCS message area
: Bei KDCS-Aufrufen: Puffer-Bereich, in dem Nachrichten oder Daten für openUTM oder für das *Teilprogramm* bereitgestellt werden.

Nachrichten-Verteiler (BS2000/OSD)
message router
: Einrichtung in einem zentralen Rechner oder Kommunikationsrechner zur Verteilung von Eingabe-Nachrichten an unterschiedliche *UTM-Anwendungen*, die auf unterschiedlichen Rechnern liegen können. Der Nachrichten-Verteiler ermöglicht außerdem, mit *Multiplexanschlüssen* zu arbeiten.

Network File System/Service / NFS
: Ermöglicht den Zugriff von Unix-Rechnern auf Dateisysteme über das Netzwerk.

Netzprozess (Unix-/Windows-Systeme)
net process
: Prozess einer *UTM-Anwendung* zur Netzanbindung.

Netzwerk-Selektor
network selector
: Der Netzwerk-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Vermittlungsschicht des *OSI-Referenzmodells*.

Normale Beendigung einer UTM-Anwendung
normal termination of a UTM application
: Kontrollierte Beendigung einer *UTM-Anwendung*; das bedeutet u.a., dass die Verwaltungsdaten auf der *KDCFILE* aktualisiert werden. Eine normale Beendigung veranlasst der *Administrator* (z.B. mit KDCSHUT N). Den Start nach einer normalen Beendigung führt openUTM als *Kaltstart* durch.

Object Identifier
object identifier
: Ein Object Identifier ist ein global (d.h. weltweit) eindeutiger Bezeichner für Objekte im OSI-Umfeld. Ein Object Identifier besteht aus einer Folge von ganzen Zahlen, die einen Pfad in einer Baumstruktur repräsentiert.

Offener Terminalpool
open terminal pool
: *Terminalpool*, der nicht auf *Clients* eines Rechners oder eines bestimmten Typs beschränkt ist. An diesen Terminalpool können sich alle Clients anschließen, für die kein Rechner- oder Typ-spezifischer Terminalpool generiert ist.

Online-Import
online import
: Als Online-Import wird in einer *UTM-Cluster-Anwendung* das Importieren von Anwendungsdaten aus einer normal beendeten Knoten-Anwendung in eine laufende Knoten-Anwendung bezeichnet.

Online-Update
online update
: Als Online-Update wird in einer *UTM-Cluster-Anwendung* die Änderung der Konfiguration der Anwendung oder des Anwendungsprogramms oder der Einsatz einer neuen UTM-Korrekturstufe bei laufender *UTM-Cluster-Anwendung* bezeichnet.

openSM2
: Die Produktlinie openSM2 ist eine einheitliche Lösung für das unternehmensweite Performance Management von Server- und Speichersystemen. openSM2 bietet eine Messdatenerfassung, Online-Überwachung und Offline-Auswertung.

openUTM-Anwendung
openUTM application
: siehe *UTM-Anwendung*.

openUTM-Cluster
openUTM cluster
: aus der Sicht von UPIC-Clients, **nicht** aus Server-Sicht:
Zusammenfassung mehrerer Knoten-Anwendungen einer UTM-Cluster-Anwendung zu einer logischen Anwendung, die über einen gemeinsamen Symbolic Destination Name adressiert wird.

openUTM-D
: openUTM-D (openUTM-Distributed) ist eine openUTM-Komponente, die *verteilte Verarbeitung* ermöglicht. openUTM-D ist integraler Bestandteil von openUTM.

OSI-LPAP-Bündel
OSI-LPAP bundle
: *LPAP-Bündel* für *OSI TP*-Partner-Anwendungen.

OSI-LPAP-Partner
OSI-LPAP partner

OSI-LPAP-Partner sind die bei openUTM generierten Adressen der *OSI TP-Partner*. Für die *verteilte Verarbeitung* über das Protokoll *OSI TP* muss in der lokalen Anwendung für jede Partner-Anwendung ein OSI-LPAP-Partner konfiguriert werden. Der OSI-LPAP-Partner spiegelt in der lokalen Anwendung die Partner-Anwendung wider. Bei der Kommunikation wird die Partner-Anwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten OSI-LPAP-Partners angesprochen.

OSI-Referenzmodell
OSI reference model

Das OSI-Referenzmodell stellt einen Rahmen für die Standardisierung der Kommunikation von offenen Systemen dar. ISO, die Internationale Organisation für Standardisierung, hat dieses Modell im internationalen Standard ISO IS7498 beschrieben. Das OSI-Referenzmodell unterteilt die für die Kommunikation von Systemen notwendigen Funktionen in sieben logische Schichten. Diese Schichten haben jeweils klar definierte Schnittstellen zu den benachbarten Schichten.

OSI TP

Von der ISO definiertes Kommunikationsprotokoll für die verteilte Transaktionsverarbeitung. OSI TP steht für Open System Interconnection Transaction Processing.

OSI TP-Partner
OSI TP partner

Partner der UTM-Anwendung, der mit der UTM-Anwendung über das OSI TP-Protokoll kommuniziert.
Beispiele für solche Partner sind:

- eine UTM-Anwendung, die über OSI TP kommuniziert
- eine Anwendung im IBM-Umfeld (z.B. CICS), die über openUTM-LU62 angeschlossen ist
- eine Anwendung des Trägersystems OpenCPIC des openUTM-Client
- Anwendungen anderer TP-Monitore, die OSI TP unterstützen

Outbound-Conversation (CPI-C)
outbound conversation

siehe *Outgoing-Conversation*.

Outgoing-Conversation (CPI-C)
outgoing conversation

Eine Conversation, bei der das lokale CPI-C-Programm der *Initiator* ist, heißt Outgoing-Conversation. In der X/Open-Specification wird für Outgoing-Conversation auch das Synonym Outbound-Conversation verwendet.

Pagepool
page pool

Teil der *KDCFILE*, in dem Anwenderdaten gespeichert werden, z.B. *Dialog-Nachrichten*, Nachrichten an *Message Queues*, *Sekundärspeicherbereiche*.

Parameterbereich
parameter area

Datenstruktur, in der ein *Teilprogramm* bei einem UTM-Aufruf die für diesen Aufruf notwendigen Operanden an openUTM übergibt.

Postselection (BS2000/OSD)
postselection

Auswahl der protokollierten UTM-Ereignisse aus der SAT-Protokolldatei, die ausgewertet werden sollen. Die Auswahl erfolgt mit Hilfe des Tools SATUT.

Prädialog (BS2000/OSD)
predialog

Aufforderung des Terminal-Benutzers an das Datenkommunikationssystem zum Aufbau einer *logischen Verbindung* zur *Anwendung*. Der Prädialog entfällt, wenn die logische Verbindung auf Initiative der Anwendung aufgebaut wird.

Preselection (BS2000/OSD)
preselection

Festlegung der für die *SAT-Beweissicherung* zu protokollierenden UTM-Ereignisse. Die Preselection erfolgt durch die UTM-SAT-Administration. Man unterscheidet Ereignis-spezifische, Benutzer-spezifische und Auftrags-(TAC-)spezifische Preselection.

Presentation-Selektor
presentation selector

Der Presentation-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Darstellungsschicht des *OSI-Referenzmodells*.

Primärspeicherbereich
primary storage area

Bereich im Arbeitsspeicher, auf den das *KDCS-Teilprogramm* direkt zugreifen kann, z.B. *Standard Primärer Arbeitsbereich*, *Kommunikationsbereich*.

Printerprozess (Unix-Systeme)
printer process
siehe *Druckerprozess*.

Programmschnittstelle zur Administration
program interface for administration
UTM-Programmschnittstelle, mit deren Hilfe der Anwender eigene *Administrationsprogramme* erstellen kann. Die Programmschnittstelle zur Administration bietet u.a. Funktionen zur *dynamischen Konfigurierung*, zur Modifikation von Eigenschaften und Anwendungsparametern und zur Abfrage von Informationen zur *Konfiguration* und zur aktuellen Auslastung der Anwendung.

Prozess
process
In den openUTM-Handbüchern wird der Begriff „Prozess“ als Oberbegriff für Prozess (Unix-/Windows-Systeme) und Task (BS2000/OSD) verwendet.

Queue
queue
siehe *Message Queue*

Quick Start Kit
Beispielanwendung, die mit openUTM (Windows-Systeme) ausgeliefert wird.

Quittungs-Auftrag
confirmation job
Bestandteil eines *Auftrags-Komplexes*, worin der Quittungs-Auftrag dem *Basis-Auftrag* zugeordnet ist. Es gibt positive und negative Quittungsaufträge. Bei positivem Ergebnis des *Basis-Auftrags* wird der positive Quittungs-Auftrag wirksam, sonst der negative.

Redelivery
redelivery
Erneutes Zustellen einer *Asynchron-Nachricht*, nachdem diese nicht ordnungsgemäß verarbeitet werden konnte, z.B. weil die *Transaktion* zurückgesetzt oder der *Asynchron-Vorgang* abnormal beendet wurde. Die Nachricht wird wieder in die Message Queue eingereiht und lässt sich damit erneut lesen und/oder verarbeiten.

Reentrant-fähiges Programm
reentrant program
Programm, dessen Code durch die Ausführung nicht verändert wird.
Im BS2000/OSD ist dies Voraussetzung dafür, *Shared Code* zu nutzen.

Request
request

Anforderung einer *Service-Funktion* durch einen *Client* oder einen anderen Server.

Requestor
requestor

In XATMI steht der Begriff Requestor für eine Anwendung, die einen Service aufruft.

Resource Manager
resource manager

Resource Manager (RMs) verwalten Datenressourcen. Ein Beispiel für RMs sind Datenbank-Systeme. openUTM stellt aber auch selbst Resource Manager zur Verfügung, z.B. für den Zugriff auf *Message Queues*, lokale Speicherbereiche und Logging-Dateien. Anwendungsprogramme greifen auf RMs über RM-spezifische Schnittstellen zu. Für Datenbank-Systeme ist dies meist SQL, für die openUTM-RMs die Schnittstelle KDCS.

RFC1006

Von IETF (Internet Engineering Task Force) definiertes Protokoll der TCP/IP-Familie zur Realisierung der ISO-Transportdienste (Transportklasse 0) auf TCP/IP-Basis.

RSA

Abkürzung für die Erfinder des RSA-Verschlüsselungsverfahrens Rivest, Shamir und Adleman. Bei diesem Verfahren wird ein Schlüsselpaar verwendet, das aus einem öffentlichen und einem privaten Schlüssel besteht. Eine Nachricht wird mit dem öffentlichen Schlüssel verschlüsselt und kann nur mit dem privaten Schlüssel entschlüsselt werden. Das RSA-Schlüsselpaar wird von der UTM-Anwendung erzeugt.

SAT-Beweissicherung (BS2000/OSD)
SAT audit

Beweissicherung durch die Komponente SAT (Security Audit Trail) des BS2000-Softwareproduktes SECOS.

Sekundärspeicherbereich
secondary storage area
Transaktionsgesicherter Speicherbereich auf der *KDCFILE*, auf den das KDCS-*Teilprogramm* mit speziellen Aufrufen zugreifen kann. Lokale Sekundärspeicherbereiche (LSSB) sind einem *Vorgang* zugeordnet, auf globale Sekundärspeicherbereiche (GSSB) kann von allen Vorgängen einer *UTM-Anwendung* zugegriffen werden. Weitere Sekundärspeicherbereiche sind der *Terminal-spezifische Langzeitspeicher (TLS)* und der *User-spezifische Langzeitspeicher (ULS)*.

Selektor
selector
Ein Selektor identifiziert im lokalen System einen *Zugriffspunkt* auf die Dienste einer Schicht des *OSI-Referenzmodells*. Jeder Selektor ist Bestandteil der Adresse des Zugriffspunktes.

Semaphor (Unix-/Windows-Systeme)
semaphore
Betriebsmittel auf Unix- und Windows-Systemen, das zur Steuerung und Synchronisation von Prozessen dient.

Server
server
Ein Server ist eine *Anwendung*, die *Services* zur Verfügung stellt. Oft bezeichnet man auch den Rechner, auf dem Server-Anwendungen laufen, als Server.

Server-Seite einer Conversation (CPI-C)
server side of a conversation
Begriff ersetzt durch *Akzeptor*.

Server-Server-Kommunikation
server-server communication
siehe *verteilte Verarbeitung*.

Service Access Point
siehe *Dienstzugriffspunkt.*

Service
service
Services bearbeiten die *Aufträge*, die an eine Server-Anwendung geschickt werden. Ein Service in einer UTM-Anwendung wird auch *Vorgang* genannt und setzt sich aus einer oder mehreren *Transaktionen* zusammen. Ein Service wird über den *Vorgangs-TAC* aufgerufen. Services können von *Clients* oder anderen Services angefordert werden.

Service-gesteuerte Queue
service controlled queue
: *Message Queue*, bei der der Abruf und die Weiterverarbeitung der Nachrichten durch *Services* gesteuert werden. Ein Service muss zum Lesen der Nachricht explizit einen KDCS-Aufruf (DGET) absetzen.
Service-gesteuerte Queues gibt es bei openUTM in den Varianten *USER-Queue*, *TAC-Queue* und *Temporäre Queue*.

Service Routine
service routine
: siehe *Teilprogramm*.

Session
session
: Kommunikationsbeziehung zweier adressierbarer Einheiten im Netz über das SNA-Protokoll *LU6.1*.

Session-Selektor
session selector
: Der Session-Selektor identifiziert im lokalen System einen *Zugriffspunkt* zu den Diensten der Kommunikationssteuerschicht (Session-Layer) des *OSI-Referenzmodells*.

Shared Code (BS2000/OSD)
shared code
: Code, der von mehreren Prozessen gemeinsam benutzt werden kann.

Shared Memory
shared memory
: Virtueller Speicherbereich, auf den mehrere Prozesse gleichzeitig zugreifen können.

Shared Objects (Unix-/Windows-Systeme)
shared objects
: Teile des *Anwendungsprogramms* können als Shared Objects erzeugt werden. Diese werden dynamisch zur Anwendung dazugebunden und können im laufenden Betrieb ausgetauscht werden. Shared Objects werden mit der KDCDEF-Anweisung SHARED-OBJECT definiert.

Sicherungspunkt
synchronization point, consistency point
Ende einer *Transaktion*. Zu diesem Zeitpunkt werden alle in der Transaktion vorgenommenen Änderungen der *Anwendungsinformation* gegen Systemausfall gesichert und für andere sichtbar gemacht. Während der Transaktion gesetzte Sperren werden wieder aufgehoben.

single system image
Unter single system image versteht man die Eigenschaft eines *Clusters*, nach außen hin als eine einziges, in sich geschlossenes System zu erscheinen. Die heterogene Natur des Clusters und die interne Verteilung der Ressourcen im Cluster ist für die Benutzer des Clusters und die Anwendungen, die mit dem Cluster kommunizieren, nicht sichtbar.

SOA
SOA (Service-oriented architecture).
Eine SOA ist ein Konzept für eine Systemarchitektur, in dem Funktionen in Form von wieder verwendbaren, technisch voneinander unabhängigen und fachlich lose gekoppelten *Services* implementiert werden. Services können unabhängig von zugrunde liegenden Implementierungen über Schnittstellen aufgerufen werden, deren Spezifikationen öffentlich und damit vertrauenswürdig sein können. Service-Interaktion findet über eine dafür vorgesehene Kommunikationsinfrastruktur statt.

SOAP
SOAP (Simple Object Access Protocol) ist ein Protokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können. SOAP stützt sich auf die Dienste anderer Standards, XML zur Repräsentation der Daten und Internet-Protokolle der Transport- und Anwendungsschicht zur Übertragung der Nachrichten.

Socket-Verbindung
socket connection
Transportsystem-Verbindung, die die Socket-Schnittstelle verwendet. Die Socket-Schnittstelle ist eine Standard-Programmschnittstelle für die Kommunikation über TCP/IP.

stand-alone Anwendung
stand-alone application
siehe *stand-alone UTM-Anwendung*.

stand-alone UTM-Anwendung
stand-alone UTM application
: Herkömmliche *UTM-Anwendung*, die nicht Bestandteil einer *UTM-Cluster-Anwendung* ist.

Standard Primärer Arbeitsbereich/SPAB (KDCS)
standard primary working area
: Bereich im Arbeitsspeicher, der jedem KDCS-*Teilprogramm* zur Verfügung steht. Sein Inhalt ist zu Beginn des Teilprogrammlaufs undefiniert oder mit einem Füllzeichen vorbelegt.

Startformat
start format
: Format, das openUTM am Terminal ausgibt, wenn sich ein Benutzer erfolgreich bei der *UTM-Anwendung* angemeldet hat (ausgenommen nach *Vorgangs-Wiederanlauf* und beim *Anmelde-Vorgang*).

statische Konfigurierung
static configuration
: Festlegen der *Konfiguration* bei der Generierung mit Hilfe des UTM-Tools *KDCDEF*.

SYSLOG-Datei
SYSLOG file
: siehe *System-Protokolldatei*.

System-Protokolldatei
system log file
: Datei oder Dateigeneration, in die openUTM während des Laufs einer *UTM-Anwendung* alle UTM-Meldungen protokolliert, für die das *Meldungsziel* SYSLOG definiert ist.

TAC
TAC
: siehe *Transaktionscode*.

TAC-Queue
TAC queue
: *Message Queue*, die explizit per KDCDEF-Anweisung generiert wird. Eine TAC-Queue ist eine *Service-gesteuerte Queue* und kann unter dem generierten Namen von jedem Service aus angesprochen werden.

Teilprogramm
program unit

UTM-*Services* werden durch ein oder mehrere Teilprogramme realisiert. Die Teilprogramme sind Bestandteile des *Anwendungsprogramms*. Sie enthalten UTM-Funktionsaufrufe und sind über *Transaktionscodes* ansprechbar. Einem Teilprogramm können mehrere Transaktionscodes zugeordnet werden.

Temporäre Queue
temporary queue

Message Queue, die dynamisch per Programm erzeugt wird und auch wieder per Programm gelöscht werden kann, vgl. *Service-gesteuerte Queue*.

Terminal-spezifischer Langzeitspeicher/TLS (KDCS)
terminal-specific long-term storage

Sekundärspeicher, der einem *LTERM-, LPAP-* oder *OSI-LPAP-Partner* zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

Timerprozess (Unix-/Windows-Systeme)
timer process

Prozess, der Aufträge zur Zeitüberwachung von *Workprozessen* entgegennimmt, sie in ein Auftragsbuch einordnet und nach einer im Auftragsbuch festgelegten Zeit den Workprozessen zur Bearbeitung wieder zustellt.

TNS (Unix-/Windows-Systeme)

Abkürzung für den Transport Name Service, der einem Anwendungsnamen einen Transport-Selektor und das Transportsystem zuordnet, über das die Anwendung erreichbar ist.

Tomcat

siehe *Apache Tomcat*

Transaktion
transaction

Verarbeitungsabschnitt innerhalb eines *Services*, für den die Einhaltung der *ACID-Eigenschaften* garantiert wird. Von den in einer Transaktion beabsichtigten Änderungen der *Anwendungsinformation* werden entweder alle konsistent durchgeführt oder es wird keine durchgeführt (Alles-oder-Nichts Regel). Das Transaktionsende bildet einen *Sicherungspunkt*.

Transaktionscode/TAC
transaction code
: Name, über den ein *Teilprogramm* aufgerufen werden kann. Der Transaktionscode wird dem Teilprogramm bei der *statischen* oder *dynamischen Konfigurierung* zugeordnet. Einem Teilprogramm können auch mehrere Transaktionscodes zugeordnet werden.

Transaktionsrate
transaction rate
: Anzahl der erfolgreich beendeten *Transaktionen* pro Zeiteinheit.

Transfer-Syntax
transfer syntax
: Bei *OSI TP* werden die Daten zur Übertragung zwischen zwei Rechnersystemen von der lokalen Darstellung in die Transfer-Syntax umgewandelt. Die Transfer-Syntax beschreibt die Daten in einem neutralen Format, das von allen beteiligten Partnern verstanden wird. Jeder Transfer-Syntax muss ein *Object Identifier* zugeordnet sein.

Transport-Selektor
transport selector
: Der Transport-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Transportschicht des *OSI-Referenzmodells*.

Transportsystem-Anwendung
transport system application
: Anwendung, die direkt auf der Transportsystem-Schnittstelle wie z.B. CMX oder Socket aufsetzt. Für den Anschluss von Transportsystem-Anwendungen muss bei der Konfigurierung als Partnertyp APPLI oder SOCKET angegeben werden. Eine Transportsystem-Anwendung kann nicht in eine globale Transaktionssicherung eingebunden werden.

TS-Anwendung
TS application
: siehe *Transportsystem-Anwendung.*

Typisierter Puffer (XATMI)
typed buffer
: Puffer für den Austausch von typisierten und strukturierten Daten zwischen Kommunikationspartnern. Durch diese typisierten Puffer ist die Struktur der ausgetauschten Daten den Partnern implizit bekannt.

UPIC

Trägersystem für UTM-Clients. UPIC steht für Universal Programming Interface for Communication.

UPIC-Client

Bezeichnung für UTM-Clients mit Trägersystem UPIC

USER-Queue

USER queue

Message Queue, die openUTM jeder Benutzerkennung zur Verfügung stellt. Eine USER-Queue zählt zu den *Service-gesteuerten Queues* und ist immer der jeweiligen Benutzerkennung zugeordnet. Der Zugriff von fremden UTM-Benutzern auf die eigene USER-Queue kann eingeschränkt werden.

User-spezifischer Langzeitspeicher/ULS

user-specific long-term storage

Sekundärspeicher, der einer *Benutzerkennung,* einer *Session* oder einer *Association* zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

USLOG-Datei

USLOG file

siehe *Benutzer-Protokolldatei.*

UTM-Anwendung

UTM application

Eine UTM-Anwendung stellt *Services* zur Verfügung, die Aufträge von *Clients* oder anderen Anwendungen bearbeiten. openUTM übernimmt dabei u.a. die Transaktionssicherung und das Management der Kommunikations- und Systemressourcen. Technisch gesehen ist eine UTM-Anwendung eine Prozessgruppe, die zur Laufzeit eine logische Server-Einheit bildet.

UTM-Cluster-Anwendung

UTM cluster application

UTM-Anwendung, die für den Einsatz auf einem Cluster generiert ist und die man logisch als **eine** Anwendung betrachten kann.
Physikalisch gesehen besteht eine UTM-Cluster-Anwendung aus mehreren, identisch generierten UTM-Anwendungen, die auf den einzelnen *Knoten* laufen.

UTM-D

siehe *openUTM-D.*

UTM-Datenstation

UTM terminal

Begriff ersetzt durch *LTERM-Partner.*

UTM-F

UTM-Anwendungen können als UTM-F-Anwendungen (UTM-Fast) generiert werden. Bei UTM-F wird zugunsten der Performance auf Platteneingaben/-ausgaben verzichtet, mit denen bei *UTM-S* die Sicherung von Benutzer- und Transaktionsdaten durchgeführt wird. Gesichert werden lediglich Änderungen der Verwaltungsdaten.

UTM-gesteuerte Queues

UTM controlled queue

Message Queues, bei denen der Abruf und die Weiterverarbeitung der Nachrichten vollständig durch openUTM gesteuert werden. Siehe auch *Asynchron-Auftrag, Hintergrund-Auftrag* und *Asynchron-Nachricht*.

UTM-S

Bei UTM-S-Anwendungen sichert openUTM neben den Verwaltungsdaten auch alle Benutzerdaten über ein Anwendungsende und einen Systemausfall hinaus. Außerdem garantiert UTM-S bei allen Störungen die Sicherheit und Konsistenz der Anwendungsdaten. Im Standardfall werden UTM-Anwendungen als
UTM-S-Anwendungen (UTM-Secure) generiert.

UTM-SAT-Administration (BS2000/OSD)

UTM SAT administration

Durch die UTM-SAT-Administration wird gesteuert, welche sicherheitsrelevanten UTM-Ereignisse, die im Betrieb der *UTM-Anwendung* auftreten, von *SAT* protokolliert werden sollen. Für die UTM-SAT-Administration wird eine besondere Berechtigung benötigt.

UTM-Seite

UTM page

Ist eine Speichereinheit, die entweder 2K oder 4K umfasst. Die Größe wird bei der Generierung der *UTM-Anwendung* festgelegt. *Pagepool* und Wiederanlauf-Bereich der KDCFILE werden in Einheiten der Größe einer UTM-Seite unterteilt.

utmpfad (Unix-/Windows-Systeme)

utmpath

Das Dateiverzeichnis unter dem die Komponenten von openUTM installiert sind, wird in diesem Handbuch als *utmpfad* bezeichnet.
Um einen korrekten Ablauf von openUTM zu garantieren, muss die Umgebungsvariable UTMPATH auf den Wert von *utmpfad* gesetzt werden. Auf Unix-Systemen müssen Sie UTMPATH vor dem Starten einer UTM-Anwendung setzen, auf Windows-Systemen wird UTMPATH bei der Installation gesetzt.

Verarbeitungsschritt
processing step
Ein Verarbeitungsschritt beginnt mit dem Empfangen einer *Dialog-Nachricht*, die von einem *Client* oder einer anderen Server-Anwendung an die *UTM-Anwendung* gesendet wird. Der Verarbeitungsschritt endet entweder mit dem Senden einer Antwort und beendet damit auch den *Dialog-Schritt* oder er endet mit dem Senden einer Dialog-Nachricht an einen Dritten.

Verbindungsbündel
connection bundle
siehe *LTERM-Bündel*.

Verschlüsselungsstufe
encryption level
Die Verschlüsselungsstufe legt fest, ob und inwieweit ein Client Nachrichten und Passwort verschlüsseln muss.

Verteilte Transaktion
distributed transaction
siehe *globale Transaktion*.

Verteilte Transaktionsverarbeitung
Distributed Transaction Processing
Verteilte Verarbeitung mit *globalen Transaktionen*.

Verteilte Verarbeitung
distributed processing
Bearbeitung von *Dialog-Aufträgen* durch mehrere Anwendungen oder Übermittlung von *Hintergrundaufträgen* an eine andere Anwendung. Für die verteilte Verarbeitung werden die höheren Kommunikationsprotokolle *LU6.1* und *OSI TP* verwendet. Über openUTM-LU62 ist verteilte Verarbeitung auch mit LU6.2 Partnern möglich. Man unterscheidet verteilte Verarbeitung mit *globalen Transaktionen* (Anwendungs-übergreifende Transaktionssicherung) und verteilte Verarbeitung ohne globale Transaktionen (nur lokale Transaktionssicherung). Die verteilte Verarbeitung wird auch Server-Server-Kommunikation genannt.

Vorgang (KDCS)
service
Ein Vorgang dient zur Bearbeitung eines *Auftrags* in einer *UTM-Anwendung*. Er setzt sich aus einer oder mehreren *Transaktionen* zusammen. Die erste Transaktion wird über den *Vorgangs-TAC* aufgerufen. Es gibt *Dialog-Vorgänge* und *Asynchron-Vorgänge*. openUTM stellt den Teilprogrammen eines Vorgangs gemeinsame Datenbereiche zur Verfügung. Anstelle des Begriffs Vorgang wird häufig auch der allgemeinere Begriff *Service* gebraucht.

Vorgangs-Kellerung (KDCS)
service stacking

Ein Terminal-Benutzer kann einen laufenden *Dialog-Vorgang* unterbrechen und einen neuen Dialog-Vorgang einschieben. Nach Beendigung des eingeschobenen *Vorgangs* wird der unterbrochene Vorgang fortgesetzt.

Vorgangs-Kettung (KDCS)
service chaining

Bei Vorgangs-Kettung wird nach Beendigung eines *Dialog-Vorgangs* ohne Angabe einer *Dialog-Nachricht* ein Folgevorgang gestartet.

Vorgangs-TAC (KDCS)
service TAC

Transaktionscode, mit dem ein *Vorgang* gestartet wird.

Vorgangs-Wiederanlauf (KDCS)
service restart

Nach einem Vorgangs-Abbruch, z.B. infolge Abmeldens des Terminal-Benutzers oder Beendigung der *UTM-Anwendung,* führt openUTM einen Vorgangs-Wiederanlauf durch. Ein *Asynchron-Vorgang* wird neu gestartet oder beim zuletzt erreichten *Sicherungspunkt* fortgesetzt, ein *Dialog-Vorgang* wird beim zuletzt erreichten Sicherungspunkt fortgesetzt. Für den Terminal-Benutzer wird der Vorgangs-Wiederanlauf eines Dialog-Vorgangs als *Bildschirm-Wiederanlauf* sichtbar.

Warmstart
warm start

Starten einer *UTM-Anwendung* nach einer *abnormalen Beendigung*. Beim Warmstart wird die *KDCFILE* wieder in einen konsistenten Zustand gebracht.

Web Service
web service

Anwendung, die auf einem Web-Server läuft und über eine standardisierte und programmatische Schnittstelle (öffentlich) verfügbar ist. Die Web Services-Technologie ermöglicht es, UTM-Teilprogramme für moderne Web-Client-Anwendungen verfügbar zu machen, unabhängig davon, in welcher Programmiersprache sie entwickelt wurden.

Wiederanlauf
restart

siehe *Anwendungs-Wiederanlauf*,
siehe *Bildschirm-Wiederanlauf*,
siehe *Vorgangs-Wiederanlauf.*

Workprozess (Unix-/Windows-Systeme)
work process

Prozess, in dem die *Services* der *UTM-Anwendung* ablaufen.

WS4UTM

WS4UTM (**W**eb**S**ervices for open**UTM**) ermöglicht es Ihnen, auf komfortable Weise einen Service einer UTM-Anwendung als Web Service zur Verfügung zu stellen.

XATMI

XATMI (X/Open Application Transaction Manager Interface) ist eine von X/Open standardisierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen.
Das in openUTM implementierte XATMI genügt der XATMI CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung. XATMI in openUTM kann über die Protokolle OSI TP, *LU6.1* und UPIC kommunizieren.

XHCS (BS2000/OSD)

XHCS (Extended Host Code Support) ist ein BS2000/OSD-Softwareprodukt für die Unterstützung internationaler Zeichensätze.

XML

XML (eXtensible Markup Language) ist eine vom W3C (WWW-Konsortium) genormte Metasprache, in der Austauschformate für Daten und zugehörige Informationen definiert werden können.

Zeitgesteuerter Auftrag
time-driven job

Auftrag, der von openUTM bis zu einem definierten Zeitpunkt in einer *Message Queue* zwischengespeichert und dann an den Empfänger weitergeleitet wird. Empfänger kann sein: ein *Asynchron-Vorgang* der selben Anwendung, eine *TAC-Queue*, eine Partner-Anwendung, ein Terminal oder ein Drucker.
Zeitgesteuerte Aufträge können nur von KDCS-*Teilprogrammen* erteilt werden.

Zugangskontrolle
system access control

Prüfung durch openUTM, ob eine bestimmte *Benutzerkennung* berechtigt ist, mit der *UTM-Anwendung* zu arbeiten. Die Berechtigungsprüfung entfällt, wenn die UTM-Anwendung ohne Benutzerkennungen generiert wurde.

Zugriffskontrolle
data access control
: Prüfung durch openUTM, ob der Kommunikationspartner berechtigt ist, auf ein bestimmtes Objekt der Anwendung zuzugreifen. Die Zugriffsrechte werden als Bestandteil der Konfiguration festgelegt.

Zugriffspunkt
access point
: siehe *Dienstzugriffspunkt.*

Abkürzungen

ACSE	Association Control Service Element
AEQ	Application Entity Qualifier
AES	Advanced Encryption Standard
AET	Application Entity Title
APT	Application Process Title
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
Axis	Apache eXtensible Interaction System
BCAM	Basic Communication Access Method
BER	Basic Encoding Rules
BLS	Binder-Lader-Starter (BS2000/OSD)
CCP	Communication Control Program
CCR	Commitment, Concurrency and Recovery
CCS	Codierter Zeichensatz (Coded Character Set)
CCSN	Name des codierten Zeichensatzes (Coded Character Set Name)
CICS	Customer Information Control System (IBM)
CID	Control Identification
CMX	Communication Manager in Unix-Systemen
COM	Component Object Model
CPI-C	Common Programming Interface for Communication
CRM	Communication Resource Manager
CRTE	Common Runtime Environment (BS2000/OSD)
DB	Datenbank
DC	Data Communication
DCAM	Data Communication Access Method

DCOM	Distributed Component Object Mode
DES	Data Encryption Standard
DLM	Distributed Lock Manager (BS2000/OSD)
DMS	Data Management System
DNS	Domain Name Service
DSS	Datensichtstation (=Terminal)
DTD	Document Type Definition
DTP	Distributed Transaction Processing
DVS	Datenverwaltungssystem
EBCDIC	Binärcode für die stellenweise Verschlüsselung von Dezimalziffern (Extended Binary-Coded Decimal Interchange Code)
EJB	Enterprise JavaBeans™
FGG	File Generation Group
FHS	Format Handling System
FT	File Transfer
GSSB	Globaler Sekundärer Speicherbereich
HIPLEX®	Highly Integrated System Complex (BS2000/OSD)
HLL	High-Level Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IFG	Interaktiver Format-Generator
ILCS	Inter Language Communication Services (BS2000/OSD)
IMS	Information Management System (IBM)
IPC	Inter-Process-Communication
IRV	Internationale Referenzversion
ISO	International Organization for Standardization
J2EE	Java 2 Enterprise Edition Technologie
JCA	Java Connector Architecture
JDK	Java Development Kit
JEE5	Java Enterprise Edition 5.0
KAA	KDCS Application Area
KB	Kommunikationsbereich
KBPROG	KB-Programmbereich

KDCS	Kompatible Datenkommunikationsschnittstelle
KTA	KDCS Task Area
LAN	Local Area Network
LCF	Local Configuration File
LLM	Link and Load Module (BS2000/OSD)
LSSB	Lokaler Sekundärer Speicherbereich
LU	Logical Unit
MIGRAT	Migrationsprogramm
MQ	Message Queuing
MSCF	Multiple System Control Facility (BS2000/OSD)
NB	Nachrichtenbereich
NEA	Netzwerkarchitektur bei TRANSDATA-Systemen
NFS	Network File System/Service
NLS	Unterstützung der Landessprache (Native Language Support)
OCX	OLE Control Extension
OLTP	Online Transaction Processing
OML	Object Modul Library
OSI	Open System Interconnection
OSI TP	Open System Interconnection Transaction Processing
OSS	OSI Session Service
PCMX	Portable Communication Manager
PDN	Programmsystem für Datenfernverarbeitung und Netzsteuerung
PID	Prozess-Identifikation
PIN	Persönliche Identifikationsnummer
PLU	Primary Logical Unit
RAV	Rechenzentrums-Abrechnungs-Verfahren
RDF	Resource Definition File
RM	Resource Manager
RSA	Encryption-Algorithmus nach Rivest, Shamir, Adleman
RSO	Remote SPOOL Output (BS2000/OSD)
RTS	Runtime System (Laufzeitsystem)
SAT	Security Audit Trail (BS2000/OSD)

SECOS	Security Control System
SGML	Standard Generalized Markup Language
SLU	Secondary Logical Unit
SM2	Software Monitor 2 (BS2000/OSD)
SNA	Systems Network Architecture
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SPAB	Standard Primärer Arbeitsbereich
SQL	Structured Query Language
SSB	Sekundärer Speicherbereich
SSO	Single-Sign-On
TAC	Transaktionscode
TCEP	Transport Connection End Point
TCP/IP	Transport Control Protocol / Internet Protocol
TIAM	Terminal Interactive Access Method
TLS	Terminal-spezifischer Langzeitspeicher
TM	Transaction Manager
TNS	Transport Name Service
TP	Transaction Processing (Transaktions-Betrieb)
TPR	Task privileged (privilegierter Funktionszustand des BS2000/OSD)
TPSU	Transaction Protocol Service User
TSAP	Transport Service Access Point
TSN	Task Sequence Number
TU	Task user (nicht privilegierter Funktionszustand des BS2000/OSD)
TX	Transaction Demarcation (X/Open)
UDDI	Universal Description, Discovery and Integration
UDS	Universelles Datenbanksystem
UDT	Unstructured Data Transfer
ULS	User-spezifischer Langzeitspeicher
UPIC	Universal Programming Interface for Communication
USP	UTM-Socket-Protokoll
UTM	Universeller Transaktionsmonitor
UTM-D	UTM-Funktionen für verteilte Verarbeitung („Distributed“)

UTM-F	Schnelle UTM-Variante („Fast“)
UTM-S	UTM-Sicherheitsvariante
UTM-XML	XML-Schnittstelle von openUTM
VGID	Vorgangs-Identifikation
VTSU	Virtual Terminal Support
VTV	Verteilte Transaktionsverarbeitung
VV	Verteilte Verarbeitung
WAN	Wide Area Network
WS4UTM	WebServices for openUTM
WSDD	Web Service Deployment Descriptor
WSDL	Web Services Description Language
XA	X/Open Access Interface (Schnittstelle von X/Open zum Zugriff auf Resource Manager)
XAP	X/OPEN ACSE/Presentation programming interface
XAP-TP	X/OPEN ACSE/Presentation programming interface Transaction Processing extension
XATMI	X/Open Application Transaction Manager Interface
XCS	Cross Coupled System
XHCS	eXtended Host Code Support
XML	eXtensible Markup Language

Literatur

PDF-Dateien von allen openUTM-Handbüchern sind sowohl auf der Enterprise Edition DVD für die offenen Plattformen als auch auf der openUTM WinAdmin-DVD (für BS2000/OSD) enthalten.

Alle Handbücher sind online unter *http://manuals.ts.fujitsu.com* zu finden.

Dokumentation zu openUTM

openUTM
Konzepte und Funktionen
Benutzerhandbuch

openUTM
Anwendungen programmieren mit KDCS für COBOL, C und C++
Basishandbuch

openUTM
Einsatz von openUTM-Anwendungen unter BS2000/OSD
Benutzerhandbuch

openUTM
Einsatz von openUTM-Anwendungen unter Unix- und Windows-Systemen
Benutzerhandbuch

openUTM
Anwendungen administrieren
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose in BS2000/OSD
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose in Unix- und Windows-Systemen
Benutzerhandbuch

openUTM (BS2000/OSD, Unix-Systeme, Windows NT)
Anwendungen erstellen mit X/Open-Schnittstellen
Basishandbuch

openUTM
XML für openUTM

openUTM-Client (Unix-Systeme)
für Trägersystem OpenCPIC
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

openUTM-Client
für Trägersystem UPIC
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

openUTM WinAdmin
Grafischer Administrationsarbeitsplatz für openUTM
Beschreibung und Online-Hilfe

openUTM, **openUTM-LU62**
Verteilte Transaktionsverarbeitung
zwischen openUTM und CICS-, IMS- und LU6.2-Anwendungen
Benutzerhandbuch

openUTM (BS2000/OSD)
Anwendungen programmieren mit KDCS für Assembler
Ergänzung zum Basishandbuch

openUTM (BS2000/OSD)
Anwendungen programmieren mit KDCS für Fortran
Ergänzung zum Basishandbuch

openUTM (BS2000/OSD)
Anwendungen programmieren mit KDCS für Pascal-XT
Ergänzung zum Basishandbuch

openUTM (BS2000/OSD)
Anwendungen programmieren mit KDCS für PL/I
Ergänzung zum Basishandbuch

WS4UTM (Unix- und Windows-Systeme)
Web-Services für openUTM

openUTM
Masterindex

Dokumentation zum openSEAS-Produktumfeld

BeanConnect
Benutzerhandbuch

JConnect
Verbindung von Java-Clients zu openUTM
Benutzerdokumentation und Java-Docs

WebTransactions
Konzepte und Funktionen

WebTransactions
Template-Sprache

WebTransactions
Anschluss an openUTM-Anwendungen über UPIC

WebTransactions
Anschluss an MVS-Anwendungen

WebTransactions
Anschluss an OSD-Anwendungen

Dokumentation zum BS2000/OSD-Umfeld

Die meisten dieser Handbücher sind zusätzlich in gedruckter Form gegen gesondertes Entgelt unter *http://manualshop.ts.fujitsu.com* zu bestellen.

AID (BS2000/OSD)
Advanced Interactive Debugger
Basishandbuch
Benutzerhandbuch

BCAM (BS2000/OSD)
BCAM Band 1/2
Benutzerhandbuch

BINDER (BS2000/OSD)
Benutzerhandbuch

BS2000/OSD
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

BS2000/OSD-BC
BLSSERV
Bindelader-Starter
Benutzerhandbuch

DCAM (BS2000/OSD)
COBOL-Aufrufe
Benutzerhandbuch

DCAM (BS2000/OSD)
Makroaufrufe
Benutzerhandbuch

DCAM (BS2000/OSD)
Programmschnittstellen
Beschreibung

FHS (BS2000/OSD)
Formatierungssystem für openUTM, TIAM, DCAM
Benutzerhandbuch

IFG für FHS
Benutzerhandbuch

FHS-DOORS (BS2000/OSD, MS-Windows)
Grafische Oberfläche für BS2000/OSD-Anwendungen
Benutzerhandbuch

HIPLEX AF (BS2000/OSD)
Hochverfügbarkeit von Anwendungen in BS2000/OSD
Produkthandbuch

HIPLEX MSCF (BS2000/OSD)
BS2000-Rechner im Verbund
Benutzerhandbuch

IMON (BS2000/OSD)
Installationsmonitor
Benutzerhandbuch

MT9750 (MS Windows)
9750-Emulation unter Windows
Produkthandbuch

OMNIS/OMNIS-MENU (BS2000)
Funktionen und Kommandos
Benutzerhandbuch

OMNIS/OMNIS-MENU (BS2000)
Administration und Programmierung
Benutzerhandbuch

OMNIS-MENU (BS2000/OSD)
Benutzerhandbuch

OSS (BS2000/OSD)
OSI Session Service
User Guide

RSO (BS2000/OSD)
Remote SPOOL Output
Benutzerhandbuch

SECOS (BS2000/OSD)
Security Control System
Benutzerhandbuch

SECOS (BS2000/OSD)
Security Control System
Tabellenheft

SESAM/SQL (BS2000/OSD)
Datenbankbetrieb
Benutzerhandbuch

openSM2
Software Monitor
Band 1: Verwaltung und Bedienung

TIAM (BS2000/OSD)
Benutzerhandbuch

UDS/SQL (BS2000/OSD)
Datenbankbetrieb
Benutzerhandbuch

Unicode im BS2000/OSD
Übersichtshandbuch

VTSU (BS2000/OSD)
Virtual Terminal Support
Benutzerhandbuch

XHCS (BS2000/OSD)
8-bit-Code- und Unicode-Unterstützung im BS2000/OSD
Benutzerhandbuch

Dokumentation zum Umfeld von Unix-Systemen

CMX V6.0 (Solaris)
Betrieb und Administration
Benutzerhandbuch

CMX V6.0 (Unix-Systeme)
Betrieb und Administration
Benutzerhandbuch

CMX V6.0
CMX-Anwendungen programmieren
Programmierhandbuch

OSS (SINIX)
OSI Session Service
User Guide

PRIMECLUSTER*TM*
Konzept (Solaris, Linux)
Benutzerhandbuch

openSM2
Die Dokumentation zu openSM2 (Solaris) und openSM2 (Linux) wird in Form von ausführlichen Online-Hilfen bereitgestellt, die mit dem Produkt ausgeliefert werden.

Sonstige Literatur

CPI-C (X/Open)
Distributed Transaction Processing
X/Open CAE Specification, Version 2
ISBN 1 85912 135 7

Reference Model Version 2 (X/Open)
Distributed Transaction Processing
X/Open Guide
ISBN 1 85912 019 9

TX (Transaction Demarcation) (X/Open)
Distributed Transaction Processing
X/Open CAE Specification
ISBN 1 85912 094 6

XATMI (X/Open)
Distributed Transaction Processing
X/Open CAE Specification
ISBN 1 85912 130 6

XML
Spezifikation des W3C (www – Konsortium)
Webseite: http://www.w3.org/XML

Stichwörter

24-Bit-Adressierungsmodus 412
31-Bit-Adressierungsmodus 412
40Z 343, 362, 532
8-Bit-Code 235

A
Abbau einer Session 73
Abdruckquittung 386
Abfragen
  Druckerinformationen (RSO) 174
Ablegen, Konfigurationsinformationen 29, 31
Abrechnungsparameter, festlegen 281
Abrechnungsphase 349, 504
ABSTRACT-SYNTAX 94, 247, 273, 284
Abstrakte Syntax 93
  definieren 273
  identifizieren 536
Access Point 88
  definieren (OSI TP) 275
  generieren 96
  über AET identifizieren 553
Access-List-Konzept 216
ACCESS-POINT 94, 247, 275
ACCOUNT 245, 281
Accounting
  Abrechnungssatz 282
  CPU-Sekunde 282
  Druckausgaben 282
  Gewichtung Partner-Anwendung 349
ADABAS 233
ADD-SPOOL-CHARACTER 171
ADD-SPOOL-FORM 171
Administration
  UTM-Cluster-Anwendung über Programmschnittstelle 22
Administrations-Journal 22, 621
Administrations-TAC, generieren 489
Administrationsberechtigung 146
  Benutzer 545
  Partner-Anwendung (LU6.1) 343
  Partner-Anwendung (OSI TP) 435
  Transaktionscode 493
Administrationskommando
  asynchron, Empfänger definieren 379
  Ergebnis 379
ADMINISTRATOR=*ADMINISTRATOR 170
Administrieren
  Druckauftrags-Queue 176, 357, 445
Adresse
  generieren (OSI TP) 426
  OSI TP-Partner 88
  OSI TP-Zugriffspunkt 278
Adressformat
  ACCESS-POINT 279, 296
  BCAMAPPL 296
  CON 312
  OSI-CON 428
  PTERM 459
Adressinformation
  CMX 105
AEQ, siehe Application Entity Qualifier
AES-Verfahren 229
AES-Verschlüsselung 226
AET, siehe Application Entity Title
Aktualisieren, KDCFILE 585
Aktualisieren, KDCFILE (stand-alone UTM-Anwendung) 590

Aktualisieren, KDCFILE (UTM-Cluster-Anwendung) 595
Alias-LTERM 357
ALTERNATE-LIBRARIES 199, 338
Ändern
Pagepool-Größe 49
Verwaltungsdaten KDCFILE 48
Anmelden
automatisch 141
Anmeldeverfahren
steuern 483
announce asynchronous message 521
Anschließen
Client 125
Clientprogramm, mehrfach 130, 522
Drucker 165
Anschlusspunkt
logisch 124
Anwendung
Ablaufparameter generieren 368
dynamisch ändern 577
Locale definieren 128
Standard-Locale definieren 385
Anwendung, siehe auch UTM-Anwendung
Anwendungsglobaler Pool 202
Anwendungskontext, siehe Application Context
Anwendungslokaler Pool 202
Anwendungsmeldungsmodul 185
Anwendungsname
generieren 289
Anzahl
Asynchron-Nachrichten in Message Queue 343, 362, 436, 498
Asynchron-Vorgänge 372
automatisch aufgebauter Verbindungen 434
Benutzer/Clients 377
Drucker-Nachrichten für Verbindungsaufbau 361
Einträge TRACE-Bereich 401
erzeugbarer Namen 259
GSSBs 381
Key-/Lockcodes 384
LSSBs 387
LTERM-Partner für LTERM-Pool 530
parallele Verbindungen 433
Prozesse 400
reservierte Objekte eines Objekttyps 466
Semaphore key 397
Sessions 345
Transportverbindungen 309
unbeantwortete Teil-Nachrichten 476
UTM-Seiten des Cache-Speichers 375
UTM-Seiten für Pagepool 390
Verbindungen definieren (OSI TP) 423, 429
APDN 455
APPLI 455
Application Context 93
definieren 283
festlegen 283, 429
generieren 95
OSI-LPAP 283, 429
Application Entity Qualifier 91, 92, 429
OSI-LPAP 431
Application Entity Title 91, 276, 429, 553
generieren 95
Application Process Title 91, 429, 553
OSI-LPAP 432
UTMD 553
APPLICATION-CONTEXT 94, 247
APT, siehe Application Process Title
Arbeitsschritte
aktualisieren KDCFILE für stand-alone Anwendung 590
aktualisieren KDCFILE für UTM-Cluster-Anwendung 595
AREA 200, 244, 286
ASCII-Code 236
ASCII/EBCDIC 425, 450, 476
Association
OSI TP 90
Asynchron-Nachricht 48
anzeigen 356
Ausgabe ohne Vorankündigung (MUX) 193
maximale Anzahl 363
verwendeter Zeichensatz 240
Asynchron-Prozesse, maximale Anzahl 372
Asynchron-Verarbeitung 372
Asynchron-Vorgang 48, 505

Asynchrone Administration, FPUT 52
ASYNTASKS 372
AT-PRINTER-START 170
Aufbau
  Session 73, 476
Aufrufen
  inverser KDCDEF 267
  KDCDEF 251, 593
  KDCDEF (BS2000/OSD) 251
  KDCDEF (Unix-Systeme) 252
  KDCDEF (Windows-Systeme) 253
  KDCUPD (BS2000/OSD) 593
  KDCUPD (Unix-Systeme) 593
  KDCUPD (Windows-Systeme) 593
Aufspalten, KDCFILE 56
Auftraggeber-Anwendung 223
Auftragnehmer-Antwort, Wartezeit 353
Auftragnehmer-Anwendung 224
Auftragnehmer-Quittung
  Wartezeit 555
Auftragnehmer-Vorgänge
  maximale Anzahl 555
Auftrags-spezifische SAT-Protokollierung 501
Auftragssteuerung 205
Ausführungszeit, zeitgesteuerter Auftrag 379
Ausgabe-Nachrichten 48
Auslastung
  steuern 128, 377
Austauschen
  Shared Code 201
Ausweisinformation 540
  Länge definieren 376
Ausweisprüfung, generieren 540
Autolink-Funktion 199, 338
Automatikmodus, Drucker 176
Automatisch
  anmelden 141
  Berechtigungsprüfung 364
  KDCSIGN 141, 364
  Steueranweisungen erzeugen 265, 314
  Verbindungsaufbau 141, 167
  Vorgangswiederanlauf 363, 550

B

BADTACS 489
Basisname 45, 46
  KDCFILE 45, 383
  neue KDCFILE (KDCUPD) 594
BCAM 374
  Generierung 443
  verteilte Verarbeitung über LU6.1/OSI TP 104
BCAM-Anwendungsname 289
BCAM-Name 443
BCAMAPPL 79, 117, 245, 246, 289, 444
  für Clients 128
BCAMAPPL-Name 581
BCMAP
  für Clients 153, 157
BCMAP-Einträge 104
Beispiel
  Generierung ComfoTRAVEL 559
  Locale generieren 240
Benutzer
  Berechtigungsprüfung 544
  maximale Anzahl definieren 377
Benutzer-Protokolldatei
  doppelte 402
  Name definieren 383
  zwischengespeicherte Sätze 48
Benutzer-spezifisch
  Meldungsziele 188
  Queues 178
  SAT-Protokollierung 551
  Sprachumgebung 239
  Startformat 541
Benutzerdaten
  aktuelle 48
  codieren für Übertragung 283
  KDCFILE 29
  übertragen 93, 273
  verschlüsseln 227
Benutzerdienstprotokoll 453, 531

Benutzerkennung 31, 140, 365, 584
  definieren 538
  dynamisch ändern 577
  LTERM-Partner 364
  Passwort 544
  sperren 552
Benutzermeldungsmodul 185
  definieren 184, 407
  Sprachumgebung festlegen 407
Benutzerrollen zuordnen 219
Berechtigungsprofil, LTERM-Partner 125
Bereiche, KDCFILE 44
Beschreiben, Formatierungssystem 332
Betriebsparameter 48
Bindestrich
  in Namen 258
Blanks, führende 384
Blockierender Aufruf 509
  Wartezeit 392, 557
Blocklänge, Benutzerdaten 387
Blockung
  KDCFILE 44
  Pagepool 49
BLS
  Schnittstelle nutzen 336
BRETRYNR, MAX 374
BS2000/OSD
  Minimalkonfiguration 35
BUNDLE 134
Bypass-Betrieb 175
  Drucker 175

C

Cache 49
  (nicht) residenter 375
  Cache-Paging-Algorithmus 376
  Eigenschaften definieren 375
  Größe definieren 375
CAE-Spezifikation 469
CATID
  KDCFILE 598
catid 45
CCS-Name 237
  Editoption 325, 360, 385, 527, 543
  Editprofil 241
CERTIFICATE 541
CID 445
  Drucker 176
CIS 232, 317
PRINTER, printertype, 457
Client 124
  anschließen 125
  BCMAP-Eintrag für 153, 157
  definieren 354, 364
  definieren (PTERM) 440
  Eigenschaften für physischen 126
  logischer 125
  LTERM-Partner zuordnen 125
  mehrfach anschließen 130
  sicherer 229
  Status 458
  TS-Anwendung 455, 456
  vertrauenswürdig (trusted) 229
Client-Programm, siehe openUTM-Client-Programm
Client/Server-Verbund, generieren 160
CLUSTER 245, 247, 297
Cluster
  Knoten-Anwendung konfigurieren 306
Cluster-Administrations-Journal 621
Cluster-globale Datei
  Sperre 304
  Wiederholung der Sperranforderung 304
Cluster-interne Kommunikation
  Kommunikationsendpunkt 299
  Portnummer 299
Cluster-Konfigurationsdatei 66
  erzeugen 420, 421
CLUSTER-NODE 245, 247, 306
Cluster-User-Datei 66, 300
  für bestehende UTM-Cluster-Anwendung erzeugen 596
CMX
  Adressinformation bereitstellen 105
Code-Konvertierung
  für TS-Anwendungen 149

CODED-CHARACTER-SET 238
Coded-Character-Set-Name 237
Codetypen 349
ComfoTRAVEL 559
Commit Functional Unit 276
Common Memory Pool
  generieren 202, 411
  Größe 202
  Größe generieren 413
Compiler 437, 439
CON 79, 246, 271, 309
  dynamisch ändern 579
CONN-USERS 128, 377
connection control time 554
Connection Request 458
CONNECTION REQUEST TIME 168, 378
CONRTIME 378
CONSOLE 185
Contention Loser 475
  LU6.1 73
  OSI TP 90
Contention Winner 475, 476
  LU6.1 73
  OSI TP 90
Control Identification, Drucker 176
control_statements_file 314, 420
CONTWIN 475
CPI-C-Teilprogramme
  blockierende Aufrufe 497, 509
  TAC 489, 494
CPU-Sekunde, gewichten 282
CPU-Zeit
  Verbrauch definieren 504
CREATE-CONTROL-STATEMENTS 42, 245, 267, 314
cssname 238
CTERM 357

D

Darstellungsmittel 27
Darstellungsschicht 90
DATABASE 232, 244, 245
Dateiverzeichnis filebase, weitere Dateien 46
Datenaustausch
  mit Kommunikationspartner 95
  Transfer Syntax (BER) 93
Datenbankkopplung
  definieren 232, 317
  Multi-Instanzen-Betrieb 234
Datenbankschlüssel 494
Datenbanksystem 232, 233
  definieren 317, 469
  KDCUPD 587
  maximale Anzahl 317
Datenbereiche
  definieren 286
  shareable 203
Datenschutz 144
Datensicherheit 47
Datenübertragung, Application Context 283
DB 317
DB2 233
DCAM-Anwendung 455
dclose(3X) 480
Dead Letter Queue 488
  Anzahl Nachrichten 379
DEFAULT 244, 320
Definieren
  Abstrakte Syntax 273
  Alias-LTERM 357
  Application Context 283
  Basisname KDCFILE 383
  Benutzerkennung 538
  Benutzermeldungsmodul 184, 407
  Client 354
  Datenbankkopplung 232, 317
  Drucker 165, 354
  Druckersteuer-LTERM 176, 357
  Editoptionen 324
  Event-Exits 330
  Funktionstasten 477
  Keyset 334
  Konfiguration 251
  Locale 238
  Logische Verbindung (LU6.1) 309
  LPAP-Partner 341
  LTERM-Gruppe 357

Definieren (Forts.)
LTERM-Pool 518
Master-LPAP eines LU6.1-LPAP-Bündels 366
Master-LPAP eines OSI-LPAP-Bündels 367
Master-LTERM 356
Multiplexverbindungen 191
OSI-LAP-Partner 429
OSI-LPAP-Bündel 367
parallele Verbindungen (LU6.1) 289
physische Clients/Drucker 440
Prozesszahlen 368
Resource Manager 469
RSO-Drucker 169
Sessionname 345
Shared Objects 480
Standardwerte 320
Startformat 358
TAC 488
TAC-Klassen 506
Teilprogramm 437
Transfersyntax 536
Denkzeit, Benutzer 401
DES-Verfahren 229
DES-Verschlüsselung 226
destination_process_name 475
Device-Definition, RSO-Drucker 170
Diagnosedaten 401
Dialog-Nachrichten 48
verwendeter Zeichensatz 239
Dialog-TACs 503
Dialog-Vorgang 505
Dialogführung 557
Dienstzugriffspunkt, siehe Access Point
DISCONNECT PENDING 193
DISCONNECT=*YES 170
dlerror(3X) 480
dlopen(3X) 480
dlsym(3X) 480
Dokumentation, Wegweiser 14
Doppelte Dateiführung
generieren für KDCFILE 384
KDCFILE 44
raw-device 61
DPUT
LTERM-Bündel 133
LTERM-Gruppe 137
DPUTLIMIT1 379
DPUTLIMIT2 380
Druckauftrag
gewichten 282
Queue administrieren 176
Drucker
Bypass-Betrieb 175
Control Identification (CID) 176
definieren 165, 354, 364
definieren (PTERM) 440
Drucker-ID 167, 445
Druckerbündel generieren 175
Druckergruppe 166
dynamisch ändern 577
freigeben im Fehlerfall (RSO) 175
für openUTM aktivieren (RSO) 174
generieren 165
identifizieren 167
Informationen abfragen (RSO) 174
logischer 165
LTERM-Partner zuordnen 167
OLTP-Schnittstelle 169
Sharing 166
Status 458
Status (RSO) 174
Verbindungsaufbau 361
Drucker 9001-3 455
Drucker 9001-893 455
Drucker 9011-18 455
Drucker 9011-19 455
Drucker 9012 455
Drucker 9013 455
Drucker 9022 455
Drucker Sharing 361
Drucker-Queue, administrieren 357, 445
Druckersteuer-LTERM 176, 445
definieren 176, 357
festlegen 165
Druckersteuerung 445
Druckertyp 167
Druckprozess 446

DSS 3270 (IBM) 454
DSS 9748 454
DSS 9749 454
DSS 9750 454
DSS 9751 454
DSS 9752 454
DSS 9753 454
DSS 9754 454
DSS 9756 454
DSS 9763 454
DSS X28 (TELETYPE) 454
DSS X28 (VIDEO) 454
Dynamisch
  eintragen, Voraussetzungen 581
  Objekte eintragen 265, 464, 577
  Objekte löschen 266, 271

E

EBCDIC-Zeichensätze 236
EBCDIC/ASCII-Konvertierung 528
Echte Benutzerkennung 140
  Wiederanlauf 147
EDIT 245, 324
  CCS-Name definieren 325
  Zeichensatz definieren 325, 360, 385, 527, 543
Editoptionen, definieren 324
Editprofil
  definieren 324
  Zeichensatzname 241
Eigenschaften
  Clients und Drucker 31
  LTERM-Pools 129
  Partner-Anwendungen 31
  physische Clients 126
  TACs 31, 488
  UTM-Anwendung 31, 368
Ein-/Ausgaben, gewichten 282
Einfügen, Kommentarzeile 463
Eingeben
  Steueranweisungen 251
Einschalten
  SAT-Protokollierung 396
Eintragen
  dynamisch, UTM-Objekte 464, 578
  RSO-Geräteverwalter 170
EJECT 244, 255, 328
Encryption 225
ENCRYPTION-LEVEL
  LTERM-Pool 522
  PTERM 446
  Transktionscode 495
END 244, 329
ENTRY 232, 244
Ereignis-spezifische Preselection 471
Ergebnis
  inverser KDCDEF 268
  KDCDEF-Lauf 33, 264
Ersatzverbindung (OSI TP) 97
Erstellen, Generierungsdaten mit WinAdmin 31
Erweiterter Zeichensatz 235
Erzeugen
  Cluster-Konfigurationsdatei 420, 421
  KDCFILE 420
  neue KDCFILE 591
  ROOT-Tabellen-Source 420
  RSA-Schlüssel 421
  Steueranweisungen 42, 265, 314
Event-Exits 200
  definieren 330
  Event-Exit VORGANG 496
Event-Services 200, 330
EXIT 330, 496
Explizite Verbindungs-Benutzerkennung 140

F

F-Tasten 477
Fehlerunterlagen 607
Ferne Anwendung siehe Partnera-Anwendung
Festlegen
  Abrechnungsparameter 281
  Druckersteuer-LTERM 165
  RSO-Puffergröße 173
  SAT-Protokollierung 471
  SM2-Datenlieferung 398
  Sprachumgebung 238
  Standardmeldungsmodul 385

FHS-DOORS-Frontend 454
FILEBASE
Knoten-Anwendung 306
filebase
abgeleitete Dateien 45
Folgeprogramm 496
FORMAT 331, 541
Format
KDCDEF-Steueranweisungen 255
Namen in KDCDEF-Anweisungen 257
Formatierungssystem
beschreiben 332
FORMMODE 331
FORMSYS 244, 332
Fortsetzungszeilen, KDCDEF-Steueranweisungen 255
FPUT 52
LTERM-Bündel 133
LTERM-Gruppe 137
Füllzeichen
KB/SPAB 377
Funktionen
KDCUPD 597
Funktionstasten
definieren 477

G

Gekoppelter Betrieb 233
Generieren
Administrations-TAC 489
Adresse (OSI TP) 426
Anwendungseigenschaften 368
Ausweisprüfung 540
Client/Server-Verbund 160
Drucker 165
Druckerbündel 175
Lademodul (BLS) 336
Lademodule 196
logische Verbindung (OSI TP) 423
LU6.1-LPAP-Bündel 84
LU6.1-Verbindung 76
Maximalwerte 368
Multiplexanschluss 417
OSI-LPAP-Bündel 101
RSO-Drucker 169
shareable Module 203
UTM und BCAM 104
UTM-Anwendung 30
UTM-Cluster-Anwendung 64
UTM-Datenbankkopplung 234
UTM-Objekte 33
Verteilte Verarbeitung (LU6.1) 72
Verteilte Verarbeitung (OSI TP) 88
Generieren mit WinAdmin 31
Generierung
ändern 591
Generierungstool 243
Generierungsvarianten UTM 371
Maximalwerte überschritten 260
Generierung verteilter Anwendungen
mit WinAdmin 71
Generierungsdaten erstellen mit WinAdmin 31
Generierungshinweise
LU6.1-Protokoll 74
Generierungsschnittstelle 30
Gesperrte Betriebsmittel, Wartezeit auf 395
globale Eigenschaften
UTM-Cluster-Anwendung 297
Globale Werte
festlegen 97
verteilte Verarbeitung 553
Grace-Sign-On 484
Größe
Arbeitsbereich dyn. Datenablage (OSI TP) 390
Cache-Speicher 375
Common Memory Pool 413
KAA 264
LPUT-Puffer 386
Pagepool 46, 49, 390
Pagepool kontrollieren 363
RSO-Puffer 388, 402
SYSLOG-FGG 399
UTM-Seite 374
Vorgangs-Memory SQL-DB-System 402
Wiederanlaufbereich 46, 52, 393
Größenüberwachung, SYSLOG 399
GROUP 138

GSSB 48
maximale Anzahl definieren 381

H
Hauptdatei, KDCFILE 57
High Performance File 374
Hiperfile 374
Hohe Transaktionsrate 56
HOSTCODE 237
HOSTNAME 382
Hostname
Mapped 118
realer 118
UTM 118
HP-UX 13

I
I/O-Verhalten 56
Identifizieren
Drucker 167
Idle-Zustand 475
idle_time 475
IDLETIME 144, 435, 475
LTERM-Pool 525
IMON-Installationspfad
SESAM/SQL 318
UDS/SQL 318
Implizite Verbindungs-Benutzerkennung 140
INFO LO 235
INFORMIX 233, 469
INIT PU 235
Initialisieren
Verwaltungsdaten 48
INPUT 331
Inputdateien
angeben für KDCDEF-Lauf 420
KDCDEF 271
Instanz (OSI) 88
Intelligente Datenstation 455
Internationalisierung 235
Anwendung 407
Inverser KDCDEF 42, 265, 314
aufrufen 267
erzeugte Dateien 268
Objekttypen 265
starten 267
IP-Adresse
übernehmbar 382
IPC 382
IPCTRACE 382
ISAM/XA 233
ISO 646 236
ISO 8859 236
ISO-Zeichensätze 236
IUTMDB 317

K
K-Tasten 477
K001 358
K009 478
K012 193, 356
K017 504, 505
K020 364
K040 51
K041 51, 390
K055 504, 505
K064 455
K078 388
K101 51
K107 455
K140 192
K141 192
K492 275
KAA-Größe 264
KAASHMKEY 383
Katalogeintrag, KDCFILE 376
Katalogkennungen
Knoten-Anwendung 308
KB
Füllzeichen definieren 377
Länge definieren 383
KCMF 170
KCSMSGS 183, 186, 407
KCSMSGSE 183, 407
KDC Application Area 264
KDCAPLKS 334
KDCAPPL
SPOOLOUT=ON 166

KDCBADTC 207, 489
KDCDEF
  Anzahl Namen 259
  aufrufen 251, 593
  aufrufen (BS2000/OSD) 251
  aufrufen (Unix-Systeme) 252
  aufrufen (Windows-Systeme) 253
  Auswirkung auf Generierungsobjekte 248
  Eingabe beenden 329
  Ergebnis 33, 264
  Inputdateien 32, 420
  Inputdateien optimieren 271
  inverser KDCDEF 268
  KDCDEF-Lauf steuern 419
  Kommentarzeile 255, 256
  Meldungen 576
  Namensklasse 262
  Namenskonventionen 258
  Protokollierung 256
  Seitenvorschub 255
KDCDEF-Lauf mit WinAdmin 32
KDCDEF-Steueranweisungen
  Format 255
  Fortsetzungszeilen 255
  Übersicht 244
KDCFILE 29, 31, 44, 264
  aktualisieren 585
  aktualisieren für stand-alone UTM-Anwendung 590
  aktualisieren für UTM-Cluster-Anwendung 595
  Anzahl Dateien 57
  auf raw-device 58
  aufspalten 56
  Basisname 45
  Basisname definieren 383
  Bereiche 44
  Blockung 44, 374
  CATID 598
  Dateiverzeichnis filebase 46
  Datenübernahme steuern 602
  Datenübertragung mit KDCUPD 588
  doppelt generieren 376
  doppelte Dateiführung 44, 384
  erzeugen 251, 420
  erzeugen im laufenden Betrieb 54
  Hauptdatei 57
  Katalogeintrag 376
  Konsistenz prüfen 598
  kopieren 44
  neue erzeugen 591
  neuen Basisnamen festlegen 599
  Pagepool 48
  Performanceverbesserung 56
  splitten 56, 383
  UTM-Cluster-Anwendung 67
  Verwaltungsdaten 48
  Wiederanlaufbereich 51
KDCINF PTERM 193
KDCMMOD 183, 187
KDCMSGTC 207, 489
KDCMTXT 183
KDCPTERM 193
KDCROOT 29
KDCSGNTC 207, 489
KDCSIGN 364, 376, 540
  automatisches 141
KDCTXT 187
KDCUPD 594, 606
  Ablaufprotokoll steuern 600
  aufrufen 593
  aufrufen (BS2000/OSD) 593
  aufrufen (Unix-Systeme) 593
  aufrufen (Windows-Systeme) 593
  Datenübernahme steuern 602
  Konsistenz KDCFILE prüfen 598
  nötige Schritte 591
  RSA-Schlüssel 585
  Steueranweisungen 597
  Voraussetzungen 586
KDCUPD-Lauf mit WinAdmin 32
Kennzeichen
  Kommunikationspartner 436
Kerberos 545
  Zugangskontrolle generieren 242
KERBEROS-DIALOG 359, 525
Kerberos-Dialog durchführen 242, 359, 525
Kerberos-Principal 538

Kerberos-Principals 392, 405
Keycode 216, 334
  Maximalwert definieren 384
KEYSET 125
Keyset 334
  Benutzer 542
  definieren 334
  dynamisch ändern 577
  LTERM-Partner zuordnen 359
  LTERM-Pool 130, 526
  Partner-Anwendung 435
Knoten-Anwendung 63
  Antwortzeit 301
  Emergency-Kommando 302
  Failure-Kommando 304
  FILEBASE 306
  Katalogkennungen 308
  konfigurieren 306
  mit vorläufigen Werten generieren 68
  Rechnername 307
  USER-RESTART 69, 70, 305
  Verfügbarkeit 301
  Warmstart 305
  Wiederholung der Kommunikation 301
Kombination
  Multiplexverbindung-direkte Verbindung 194
Kommentarzeile
  einfügen 463
  KDCDEF 255, 256
Kommunikationsbereich
  Länge definieren 383
Kommunikationspartner
  definieren (PTERM) 440
  Kennzeichen 436
  Typ 460
  Typ definieren 454
Kommunikationssteuerschicht 90
Kompatible Zeichensätze 237
Komplexitätsstufe Passwort 546
Konfiguration 31
  Beispiel-Anwendung 559
  definieren 30, 251
  dynamisch ändern 577
  Knoten-Anwendung 306
  Konfigurationsinformationen ablegen 29, 31
  UTM-Cluster-Anwendung 297
Konfigurationsdatei
  UTM-Cluster-Anwendung 66
Konfigurationsdaten (KDCROOT) 29
Konvertierung 149
  auf native TCP/IP-Verbindung 528
KSET 217, 219, 245, 271, 334
  bei inversem KDCDEF 265
  dynamisch ändern 579

L

Lademodul (BLS)
  Autolink-Funktion 338
  generieren 336
  Lademodus 338
  Versionsnummer 340
Lademodule
  Bibliotheken 198
  generieren 196
Lademodus 197
  Lademodul 338
  Shared Object 480
Landessprachliche Meldungsmodule 184
  zuordnen 407
lang_id 183, 238
Länge
  Puffer 52
Langzeitspeicher, User-spezifisch 537
LEADING-SPACES 384
LEASY 232, 317
Leerlauf
  Session 475
  Verbindung (OSI TP) 435
Leerzeichen, führende 384
LIB 232
LINEMODE 331
Linux-Distribution 13
Listener-ID 116, 294
  MAX 389

LISTENER-PORT 127, 128
ACCESS-POINT 277
BCAMAPPL 290, 294
CON 311
OSI-CON 425
PTERM 449
LOAD-MODULE 197, 244, 245, 336
local-sessionname 72
LOCALE 125, 128, 239
Locale 235, 238
definieren 238
definieren für TPOOL 526
Generierungsbeispiel 240
LTERM-Partner-spezifisch 360
Standard-Sprachumgebung 385
LOCK 126
Lock-/Keycode-Konzept 216, 359, 496
Lockcode 216, 335, 348, 581
ferner Vorgang 347
LTERM-Partner 361
LTERM-Pool 527
Transaktionscode 496
Logische Abdruckquittung 168, 386
Logische Verbindung
definieren (LU6.1) 309
definieren (OSI TP) 423
wiederaufbauen 168, 378
Logischer Anschlusspunkt 124
Client/Drucker 354
Partner-Anwendung 341
Partner-Anwendung (OSI TP) 429
Logischer Client 125
Lokaler Servicename, dynamisch ändern 577
Lokaler Sessionname, definieren 345
Löschen
Objekte, dynamisch 266, 271
LPAP 76, 246
LU6.1-LPAP-Bündel 84
LPAP-Bündel
LU6.1-Protokoll 83
OSI TP-Protokoll 100
LPAP-Name 311
LPAP-Partner
definieren 341
Partner-Anwendung 311
Session-Eigenschaften zuordnen 343
LPUT-Daten 386
LPUT-Nachricht 387
LPUT-Puffer 386
LPUTLTH 387
LSES 80, 81, 246, 271, 345
dynamisch ändern 579
LSSB 48
maximale Anzahl definieren 387
LTAC 94, 223, 246, 247, 271, 347
bei inversem KDCDEF 265
dynamisch ändern 579
LTERM 125, 127, 165, 184, 218, 220, 245, 271, 354
bei inversem KDCDEF 265
dynamisch ändern 579
LTERM-Bündel 134
LTERM-Gruppe 138
Tabellenplätze reservieren 465
LTERM-Bündel
LTERM-Anweisung 134
PTERM-Anweisung 135
LTERM-Gruppe 136, 357
LTERM-Anweisung 138
PTERM-Anweisung 138
LTERM-Partner 124, 125, 134, 138, 581
Berechtigungsprofil 125
Client/Drucker zuordnen 450
dynamisch ändern 577
Eigenschaften definieren 354
Locale definieren 360
Locale definieren für LTERM-Pool 526
LTERM-Gruppe 136, 357
LTERM-Pool 129, 527
Message Queue 166, 362
Message Queue administrieren 165
Name 125, 165, 356
spezifische Sprachumgebung 240
Sprachumgebung 240
vordefinieren 354
Zugangsschutz 126
Zugriffsrechte festlegen 359
zuordnen 129

LTERM-Pool
Anschluss von Clients 131
Anzahl Clients 130
Arten von 129
definieren 518
Eigenschaften 129
LTERM-Partner definieren 527
LTERM-Partnernamen 130
max. Anzahl Clients 530
Message Queue 532, 548
Namen der LTERM-Partner 527
Typ des Kommunikationspartners definieren 532
Verschlüsselungsebene 522
Zugangsschutz 130
Zugriffsrechte 526
ltermname 125
ltermprefix 527
LU6.1
generieren 76
Generierungshinweise 74
LU6.1-LPAP-Bündel 83
generieren 84
LPAP zu Slave-LPAP 342
stand-alone Anwendung mit UTM-Cluster-Anwendung 85
LU6.1-Protokoll 72
LU6.2-Anwendung 71

M

Magnetstreifenkarte 540
Main-Routine KDCROOT 29
Mapped Hostnamen 118
Master-LPAP
eines LU6.1-LPAP-Bündels definieren 366
eines OSI-LPAP-Bündels definieren 367
Master-LTERM 133
festlegen 356
MASTER-LU61-LPAP 84, 246, 366
MASTER-OSI-LPAP 94, 101, 247, 367
MAX 116, 117, 128, 168, 169, 182, 185, 207, 244, 245, 247, 368
für Clients 128
MAX OSI-SCRATCH-AREA 95, 247
MAX OSISHMKEY 95, 247
MAX PRINCIPAL-LTH 392
MAX XAPTPSHMKEY 247
MAX-Anweisung 125, 168
Maximale Länge
physische Nachrichten 128, 168
Maximale Wartezeit 144
Anschluss CMP 388
Mehrfachanmeldung 143, 485
Mehrschritt-Transaktion, Wartezeit 401
Mehrsprachigkeit 235, 407
UTM-Teilprogramme 235
Meldungen
KDCDEF 576
KDCUPD 606
Meldungsanzeige
mit WinAdmin 188
Meldungsdefinitionsdatei 183, 186
Meldungsmodul 407, 410
Benutzer-spezifischer 183
englischer 183
erzeugen 183
Standard 185
Meldungstools 187
Meldungsziele 185
Benutzer-spezifisch 188
MESSAGE 184, 187, 244, 245, 407, 410
Message Queue 48
LTERM-Partner 166, 362
LTERM-Pool 532, 548
max. Anzahl Nachrichten 362
OSI-LPAP-Partner 436
Metasyntax 27
Minimalkonfiguration
BS2000/OSD 35
Unix-, Windows-Systeme 39
MODIFY-SPOOL-DEVICE 170
MODULE 244
MP-WAIT 388
MPOOL 202, 245, 411
MSCF 623
MSG-DEST 188, 245
MSGTAC 489

Multi-Instanzen-Betrieb,
Datenbankkopplung 234
MULTI-THREADED 389
Multi-threaded Netzanbindung 116
ACCESS-POINT 277
BCAMAPPL 294
Multiplexanschluss
Datenstau vermeiden 195
generieren 189, 417
Multiplexfunktion 189
Multiplexverbindung 194
definieren 191
MUX 192, 245, 417
Asynchron-Nachricht 193
Verbindungsabbau bestätigen 193
MUX-Verbindung, Statistik 194

N
Nachricht
führende Blanks 384
Nachrichtenwarteschlange 362
übergeben, Anzahl Versuche 374
Nachrichtenbereich
Länge definieren 388
Nachrichtenverschlüsselung
Transaktionscode 495
Nachrichtenverteiler 189
Funktionen 191
Name
ferne Anwendung (LU6.1) 311
Format in KDCDEF-Anweisungen 257
für Sessioneigenschaften 474
Maximalwerte für KDCDEF 259
Partner-Anwendung 311
Namensklasse
KDCDEF 262
Namenskonvention 258
Namensteil, frei wählbar 45
Namensvergabe 257
NB
Länge definieren 388
NEABT 453, 531
NET-ACCESS 388
Netzanbindung 116, 388
Neugenerierung, Empfehlungen 270
Nicht-privilegierte Subsysteme 201
NLS-Standard-Meldungskatalog 186
NRCONV 389
NUMBER 130

O
O2 233
Object-Identifier 283
Objektschutz 216
Objekttypen, inverser KDCDEF 265
Offene LTERM-Pools 129
Öffentlicher Schlüssel (RSA) 226
OMNIS 189, 417
OMNIS-MENU 192
ONCALL 339, 480
OpenCPIC, Zugriffsschutz 224
OpenCPIC-Client 139
openFT und WinAdmin 32
openUTM-Anwendung, siehe UTM-Anwendung
openUTM-Client-Programm 455
mehrfach anschließen über LTERM-Pool 522
OPTION 33, 44, 244, 419
OPTION DATA 268, 271, 420
Oracle 233
OSI TP 88
Verbindung generieren 423
Zugriffspunkt 425
OSI TP-Partner
Adresse 88
generieren 96
OSI-Begriffe 88
OSI-CON 94, 247, 423
OSI-LPAP 94, 247, 429
OSI-LPAP-Bündel 101
OSI-LPAP-Bündel 100, 367
generieren 101
stand-alone Anwendung mit UTM-Cluster-Anwendung 100
OSI-LPAP-Partner 124
definieren 429
Ersatzverbindungen 425
max. Anzahl asynchrone Nachrichten 436
Partner-Anwendung zuordnen 423

P
Pagepool 44, 48
Blockung 49
Eigenschaften definieren 390
Größe 46, 49
Größe abschätzen 49
Größe ändern 49
Größe festlegen 386
Größe kontrollieren 363
Warnstufen für Belegung 50, 390
PAM-Schreib-/Leseauftrag 394
Parallele Verbindungen
Anzahl generieren (OSI TP) 433
definieren (LU6.1) 289
generieren (OSI TP) 96
OSI TP allgemein 90
Transportverbindung generieren (LU6.1) 309
PARTNER 185
Partner-Anwendung
Administrationsberechtigung 342
Adresse (OSI TP) 426
definieren (LU6.1) 341
logischer Anschlusspunkt 341
lokalen TAC definieren 347
lokaler Anwendung zuordnen 309
LPAP-Partner definieren 341
Name 347
Name (LU6.1) 311
Zugriffsrechte 342, 435
Partner-Server-Anwendung 224
Partnertyp 454, 532
Passwort
Benutzerkennung 544
dunkelgesteuerte Eingabe 544
Komplexitätsstufe definieren 546
verschlüsseln 227
Passwort-Historie 486
PDN 455
PDN-Anwendung 455
Performance 56, 144, 371
CONTWIN 475
KDCFILE 56
Überwachung generieren 398
Periodic Write 48, 53
PGPOOLFS 57
PGWT
TAC-Klasse 509
Wartezeit 392
Physische Clients, Eigenschaften 126
Physische Nachricht, maximale Länge 401
Plattenpartition, abschätzen 59
PLU 73, 476
Pool
anwendungsglobaler 202
anwendungslokaler 202
Portnummer 102 105, 150
Preselection
Ereignis-spezifische 471
Werte voreinstellen 471
Presentation-Selektor 278
Primary Logical Unit 73, 476
PRINCIPAL 545
PRINCIPAL-LTH 392
print_level_number 361
PRINTER 456
printertype 457
Prioritätensteuerung 205
generieren 512
PRISMA 232, 317
Privater Schlüssel (RSA) 226
PROGRAM 200, 244, 245, 271, 437
bei inversem KDCDEF 265
dynamisch ändern 579
Tabellenplätze reservieren 465
program wait 509
Programm
Transaktionscode zuordnen 498
Programmaustausch 338, 480
Binder 439
generieren (BLS) 336
PROTECT-PW 546
Protokoll, Seitenvorschub veranlassen 328
Protokollierung
einschalten 396
KDCDEF 256
KDCUPD 588, 606
Prozessbeschränkung
TAC-Klassen 205

Prozesse
Anzahl pro TAC-Klasse 506
maximale Anzahl 400
Prozessor der Partner-Anwendung 312
Prozessorname 312, 452, 530
Prozesszahl definieren 368
PTCTIME 555
PTERM 126, 169, 245, 271, 440
bei inversem KDCDEF 265
dynamisch ändern 579
LTERM-Bündel 135
LTERM-Gruppe 138
Tabellenplätze reservieren 466
ptermname 126
PTYPE 127, 454, 532
LTERM-Bündel 135
LTERM-Gruppe 138
PUTMMUX
Protokoll 191
Verbindung 191

Q
QUEUE 181, 245
queue asynchronous message 362
queue_level_number 362, 532, 548
Queues
Benutzer-spezifisch 178
Dead Letter Queue 488
Service-gesteuert 178
temporäre 178
über TAC-Anweisung definiert 178, 505
Quittungsverfahren, Drucker 176

R
RAID 61
raw-device, KDCFILE 58
Readme-Dateien 19
realer Hostname 118
RECBUF 82
RECBUFFS= 57
Rechnername
Knoten-Anwendung 307
recipient_TPSU_title 349
Red Hat 13
Referenzcode 237
Reise-Buchungssystem
Beispielgenerierung 559
relocatable 382
REMARK 244, 255, 463
Remote Spool Output, siehe RSO
REMOTE-BUFFER-SIZE 169
remote-sessionname 72
RESERVE 244, 245, 464
Reserve-Knoten-Anwendung
generieren 68
Reservieren
leere Tabellenplätze 579
Tabellenplätze für UTM-Objekte 31, 464
Reservierte Namen
KDCDEF 257
RESET RSO-Drucker 170
RESET=*YES 170
Resource Manager 233
definieren 469
RESOURCE WAIT 395
RESTART
LTERM-Bündel 134
RFC1006 279
RFC1006-Kopplung
BCAM 104
Beispiel 160, 162
RMXA 244, 469
RMXA-Anweisung 233
Rollen zuordnen 219
ROOT 33, 244, 470
Tabellen-Source erzeugen 420
Tabellen-Source, Namen definieren 470
RSA 226
RSA-Schlüssel
erzeugen 421
übertragen in neue KDCFILE 585
RSA-Schlüssellänge 229
RSA-Schlüsselpaar 226
erzeugen 231
RSET 556

RSO 170
Device-Definition 170
Deviceeintrag 171
Drucker generieren 169, 456
Geräteverwalter eingetragen 170
Puffergröße festlegen 173
RTS-Module 338

S

SAT-Protokollierung
Benutzer-spezifisch 551
einschalten 396
festlegen 471
TAC-spezifisch 501
SATSEL 245, 471
Schicht (OSI) 88
Schlüssel
Semaphore 397
Schutz von Objekten 216
Seitenvorschub, Protokoll 328
Selektor
Aufbau 90
OSI TP 88
Semaphore 397
Service
geschützt 219
Service-gesteuerte Queue 48, 178
Anzahl der Neuzustellungen 182
generieren 178
Wartezeit beim Lesen 182
Zugriffsschutz 221
Service-Programm (remote)
Lockcode 348
TAC-Name 349
SESAM 317
SESAM/SQL 232
SESCHA 77, 246, 474
Session 190, 191
Abbau 73
Aufbau 73, 190
Aufbau initieren 476
Charakteristika 72
Dauer 73
Eigenschaften 72
Eigenschaften festlegen (LU6.1) 474
Leerlauf 475
Namen 72
Session-Eigenschaften zuordnen 343
Sessionaufbau, globale Überwachzeit 554
Sessionbelegung, Wartezeit 352
Sessionname definieren 345
Session Manager (OMNIS) 190, 417
Session-Selektor 278
SFUNC 245, 477
Shareable Datenbereich 203
Shareable Objekte
generieren 203
Shareable Programme
nicht-privilegierte Subsysteme 201
Shareable Teilprogramme 201
Shared Code, austauschen 201
Shared Memory 382
Shared Memory Segment
Schlüssel 374
SHARED OBJECT 244, 246, 480
Shared Objects
Dateiname 481
definieren 480
Versionsbezeichner 481
SHARED-OBJECT 439
SHUT 331
Sicherer Client 229
SIGN CL 235
SIGNON 245, 483, 489
single system image 20
SINGLE-THREADED 389
Single-threaded Netzanbindung 117
Slave-LTERM 133, 356
SM2
Datenlieferung festlegen 398
SNA-Sessions 72
SOCKET
Adressinformation bereitstellen 114
Socket-Anwendung 455, 456
Socket-Verbindung
EBCDIC/ASCII-Konvertierung 528
Solaris 13

SPAB
Füllzeichen definieren 377
Länge definieren 398
SPEC 233
Sperren
Benutzerkennung 552
Transaktionscode 502
Splitten, KDCFILE 56
SPOOL-Parameterdatei 170
Sprachkennzeichen, Locale 238
Sprachumgebung
Anwendung (Standard) 239, 385
Benutzer-spezifische 239
benutzerspezifische 543
festlegen 238
LTERM-Partner-spezifische 131, 240, 360, 526
LTERM-Partnerspezifische 125
Standard 239
stand-alone UTM-Anwendung 11, 20
Standard-Anwenderzeichensatz 238
Standard-Sprachumgebung 128, 239
Standardmeldungsmodul 185, 407
definieren 385
KCSMSGS 186
Standardwerte
Anwendungs-spezifisch 320
definieren 320
START 331
START-PRINTER-OUTPUT 170, 174
Starten
inverser KDCDEF 267
KDCDEF 251
Startfehlercode 58 399
Startformat 524, 541
Benutzer-spezifisch 541
definieren 358
STARTUP 338, 480
STATIC 339
STATION 185
Statisch generieren, Objektkomponenten 581
Statistik
zu MUX-Verbindungen 194
stdin 251, 252, 253
Steueranweisungen
eingeben 251
erzeugen 42, 265, 314
Format 255
KDCDEF Übersicht 244
KDCUPD 597
Reihenfolge 254
Steuern
Anwendungsauslastung 377
Auslastung 128
KDCDEF-Lauf 419
Stripe Set (Windows-Systeme) 61
Subsystem
nicht-privilegiert 201
SUSE 13
SYBASE 233
SYSDTA 251
SYSLINE 185
SYSLNK
DATABASE 318
SYSLOG 399
Größenüberwachung 399
SYSLST 185
SYSLST-Protokoll
Seitenvorschub generieren 328
SYSMSH.UTM.060.MSGFILE 183
SYSOUT 185
System-Standard-Code 237

T

T-PROT 127, 128, 279
T-Selektor
für Partner-Anwendung generieren 427
Tabellenplätze für UTM-Objekte
reservieren 31, 464, 579
TAC 179, 206, 215, 217, 220, 221, 244, 245, 271, 488
Administrationsberechtigung 493
bei inversem KDCDEF 265
definieren 488
dynamisch ändern 579
Tabellenplätze reservieren 466
TAC-Name für fernes Service-Programm 349

TAC-Klassen 503
definieren 506
Prioritäten festlegen 512
TAC-PRIORITIES 208, 245, 512
TAC-Queue 178, 179, 488
generieren 505
TAC-spezifische SAT-Protokollierung 501
TACCLASS 211, 245, 506, 512
TACUNIT 504
Task
Anzahl bei TAC-Klassen 507
maximale Anzahl 400
TASKS 508
TCB-Entries, definieren 516
TCBENTRY 244, 504, 516
TCP/IP
Port 102 105
Teilprogramm 582
definieren 437
dynamisch ändern 577
gewichten 349, 504
Temporäre Queue 178, 181
Terminal
dynamisch ändern 577
TERMINAL WAIT 401
Terminal-spezifischer Speicher 517
Terminalkennzeichen 312, 533
Kommunikationspartner 458
Terminalpool, siehe LTERM-Pool
TERMWAIT 144
terr_id 183, 238
Territorialkennzeichen
Locale 238
LTERM-Partner 360
TESTMODE 382, 401
Timer
Eingabe vom Dialog-Partner 448, 525
TLS 48, 245, 517
TLS-Block 517
Namen definieren 517
TNS-Betrieb 109
Umstellung auf KDCDEF 110
TPOOL 129, 245, 518
TPSU-title 91
TRACE-Bereich 401
TRACE=YES 175
Transaktionen
Platzbedarf im Wiederanlaufbereich 52
Transaktionscode 583
definieren für Partner-Anwendung 347
dynamisch ändern 577
Lockcode 496
Name definieren 493
Partner-Anwendung 347
sperren 502
Zugriffsschutz 496
Transaktionsgesicherte Verarbeitung 275
Transaktionsrate 56
TRANSFER-SYNTAX 95, 247, 536
Transfersyntax 93
definieren 536
TranSON 541
Transportprotokoll 453, 531
Transportquittung 386
Transportschicht 90
Transportsystem-Anwendung 124, 455, 456
Transportverbindung 72, 88, 191
definieren 309
Transportverbindung zu LU6.1
dynamisch ändern 577
TRMSGLTH 128
Trusted Client 229, 448
TS-Anwendung 124, 440
Code-Konvertierung 149
dynamisch ändern 577
mehrfach anschließen über LTERM-Pool 522
TSEL-FORMAT 127, 128
TSOS 175
Tuning, KDCFILE 56
TYP 232
Typ
Kommunikationspartner 454
Kommunikationspartner für LTERM-Pool 532
Typkennzeichen Datenbanksystem 232

U
Übernehmbare IP-Adresse 382
Übertragen
  Benutzerdaten 273
  RSA-Schlüssel in neue KDCFILE 585
Überwachzeit
  Sessionaufbau (LU6.1) 554
  Verbindungsaufbaus (OSI TP) 554
UDS 317
UDS/SQL 232
UDTAC 284
UDTCCR 285
UDTDISAC 285
UDTSEC 285
ULS 48, 245, 537
Umcodierung
  ASCII/EBCDIC 425, 476
Umsetztabellen 609
Umwandlungsdatei 119
Unix-Plattform 13
Unix-Systeme
  Minimalkonfiguration 39
Unterstrich, in Namen 258
Update, Verwaltungsdaten 53
UPIC-Client
  dynamisch ändern 577
UPIC-Client-Programm
  UPIC-L/UPIC-R 456
UPIC-R 455
  openUTM-Clientprogramm definieren 440
UPIC-Verbindung
  Benutzerabmeldung 486
USAGE 126, 127
  LTERM-Bündel 135
  LTERM-Gruppe 138
USER 178, 184, 218, 220, 222, 245, 271, 538
  bei inversem KDCDEF 265
  dynamisch ändern 579
  Tabellenplätze reservieren 466
User Locales
  Angaben in User-Datei übernehmen 305
User-Datei
  User Locales 305
  UTM-Cluster-Anwendung 66
USER-DEST-number (Benutzer-spezifische Meldungsziele) 188
USER-Queues 178
USER-RESTART, Knoten-Anwendung 305
User-spezifischer Langzeitspeicher 537
USERFORM 331
userid 45
USLOG-Datei, bei KDCUPD 588
USP (UTM-Socket-Protokoll) 148
UTM
  Objekte generieren 33
  Objekttabellen 31
  Seite 49
UTM-Accounting siehe Accounting
UTM-Anwendung
  generieren 30
  Namen definieren 368
  Zugang zu 440
UTM-Anwendung, siehe auch Anwendung
UTM-Benutzerkennung siehe Benutzerkennung
UTM-Cache 49
UTM-Cluster-Anwendung 11, 20, 63
  Administration (Programmschnittstelle) 22
  Administrations-Journal 22
  Antwortzeit 301
  Cluster-Administrations-Journal 621
  Cluster-Konfigurationsdatei 66
  Cluster-User-Datei 66
  Emergency-Kommando 302
  Failure-Kommando 304
  generieren 64
  Globale Eigenschaften definieren 297
  globale Speicherbereiche verwenden 69, 304
  KDCDEF-Anweisungen 67
  KDCFILE 67
  konfigurieren 297
  LU6.1-LPAP-Bündel 85
  neue Cluster-User-Datei erzeugen 596
  OSI-LPAP-Bündel 100
  übernehmbare IP-Adresse 382
  USER-RESTART 69, 70
  Verfügbarkeitsprüfung von Knoten 301
  Wiederholungen einer Kommunikation 301

UTM-D-Anwendung
generieren 72
UTM-Datenbankkopplung generieren 234
UTM-F-Anwendung 371
Verwaltungsdaten 48
UTM-Hostname 118
UTM-INFORMIX-Kopplung 233
UTM-Meldungen
Benutzer-spezifisch anpassen 183
UTM-Objekt, Tabellenplatz reservieren 579
UTM-Oracle-Kopplung 233
UTM-S-Anwendung 371
Verwaltungsdaten 48
UTM-SAT-Administrations-TACs
generieren 491
UTM-SAT-Administrationsberechtigung
Benutzer 545
Partner-Anwendung (LU6.1) 343
Partner-Anwendung (OSI TP) 435
Transaktionscode 501
UTM-Seite
Größe definieren 374
UTM-Socket-Protokoll (USP) 148
UTM_NET_HOSTNAME 121
UTMD 94, 95, 246, 247, 553
utmhostname 120
utmnetm 117
utmpfad 60, 119, 186, 187, 252, 410, 593, 610
utmpfad/msgdescription 186
utmpfad/sys/kcsmsgs.o 410

V

Variantennummern 236
Verbindung
definieren (LU6.1) 309
definieren (OSI TP) 96
Namen definieren (OSI TP) 433
parallele (OSI TP) 90
Transportprotokoll 453
wiederaufbauen 378
Verbindungs-Benutzerkennung 140, 365
Wiederanlauf 147
Verbindungsabbau
Bestätigung (MUX) 193
Verbindungsaufbau
automatisch 141, 445
globale Überwachzeit 554
RSO-Drucker 165
zyklisch nach Ausfall 378
Verbindungsbündel 133
Verbindungsmodul 232
Datenbanksystem 318
Verrechnungseinheiten
Anzahl 282
Verschlüsselung 225
Passwörter 227
Transaktionscode 495
Verschlüsselungsebene 228
festlegen für Clients 446
festlegen für LTERM-Pools 522
generieren 228
Verschlüsselungsverfahren 226
Versionsbezeichner, Shared Objects 481
Versionsnummer 199
Lademodul (BLS) 340
Versionswechsel
bei KDCUPD 586
Verteilte Verarbeitung
generieren (LU6.1) 72
generieren (OSI TP) 94
lokaler TAC 347
OSI TP 88
Parameter festlegen 553
Partner-Anwendung beschreiben (LU6.1) 341
Sessionname generieren (LU6.1) 345
Verbindung generieren (LU6.1) 309
Verbindung generieren (OSI TP) 423
Vertrauenswürdiger Client 229
Verwaltungsdaten 48
initialisieren 48
KDCFILE 29, 48
transaktionsbedingte Änderungen 51
Update 53
UTM-F-Anwendung 48
UTM-S-Anwendung 48
Verwaltungsinformation, Prozesszuordnung 117

Virtuelle Hostnamen
Umwandlungsdatei 119
UTM_NET_HOSTNAME 121
utmhostname 120
Voraussetzungen
für dynamisches Eintragen 577, 581
KDCUPD 586
VORGANG-Exit
dynamisch ändern 577
Vorgangsdaten 48
Vorgangsexit 496
Vorgangskellerung 389
Vorgangswiederanlauf 363, 541, 550
USER 550
VTSU-B 226, 236
VTSU-Code 173
VTSU-SEC 226
VTSU-Version 455

W
Warm-Start
Knoten-Anwendung 305
Warnstufe1 390
Wartezeit
Auftragnehmer-Antwort (KDCDEF) 353
Auftragnehmer-Quittung 555
beim Lesen aus Service-gesteuerten Queues 182
gesperrte Betriebsmittel 395
maximale 144
Mehrschritt-Transaktion 401
PGWT 392
Sessionbelegung 352
TAC-Queues 392
temporäre Queues 392
User-Queues 392
Wiederanlauf
echte Benutzerkennung 147
Verbindungs-Benutzerkennung 147
Wiederanlaufbereich 46, 51
aufspalten (KDCDEF) 393
Größe definieren 46, 393
Größe ermitteln 52
Wiederanlaufinformation 51
Wiederaufbau
logische Verbindung 168, 378
WinAdmin 31, 578
Generierung verteilter Anwendungen 71
Meldungsanzeige 188
Windows-System 13
Windows-Systeme
Minimalkonfiguration 39

X
XA 232, 317
XA-CAE 233
XA-P 233
XA-Schnittstelle 233, 469
xa_switch_t 469
xa_switch_t-Struktur 233
XAPTP Shared Memory 95, 402
XASWITCH 233
XATMI-Teilprogramme
blockierende Aufrufe 497, 510
TAC 489, 494
XATMIAC 285
XATMICCR 285
XHCS 239
Begriffsdefinitionen 236
Unterstützung 235

Z
Zeicheninterpretation 278
Zeichensatz
definieren, Editoption 325
erweiterter 235, 239
kompatibel 237
Zeichensatzname 237
Format 241
Locale 238
Zeitgesteuerter Auftrag
Ausführungszeit 379
Zertifikat 541
Zugang
UTM-Anwendung 440
Zugangscodes 334
Zugangskontrolle
mit Kerberos generieren 242

Zugangsschutz 145
durch Verschlüsselungsfunktionen 228
LTERM-Partner 126
LTERM-Pool 130
Zugriffsberechtigung 335
Common Memory Pool 411
Zugriffskontrolle 216
bei verteilter Verarbeitung 223
Zugriffspunkt, siehe Access Point
Zugriffsrechte
Client (LTERM-Pool) 130
Client/Client-Programme 526
für Benutzer definieren 542
LTERM-Partner 359
Partner-Anwendung 435
Zugriffsschutz 31, 334
durch Verschlüsselungsfunktionen 229
für Service-gesteuerte Queues 221
für Services definieren 97
LTERM-Pool 527
Transaktionscode 496
Zuordnen
Client/Drucker zu LTERM-Partner 450
Session-Eigenschaften 343